'68' MICRO JOURNAL

$2.95 USA

Australia	A $4.75	New Zealand	NZ $ 6.50
Singapore	S $9.45	Hong Kong	H $23.50
Malaysia	M $9.45	Sweden	30:-SEK

VOLUME VII ISSUE I • Devoted to the 68XX User • January 1985

"Small Computers Doing Big Things"

0 74470 12810 01

WE DON'T PLAY GAMES

X-12+ A SERIOUS COMPUTER IN A DESKTOP PACKAGE

Multiprocessor Technology - Combination of 8, 16 and 32 bit types

1.0 Megabyte Memory - Insures no limitation on programs

"Winchester" Disk System - Fast response, large storage capacity

UniFlex* Operating System - The standard of comparison

Hardware Floating Point - Unmatched speed in a small system

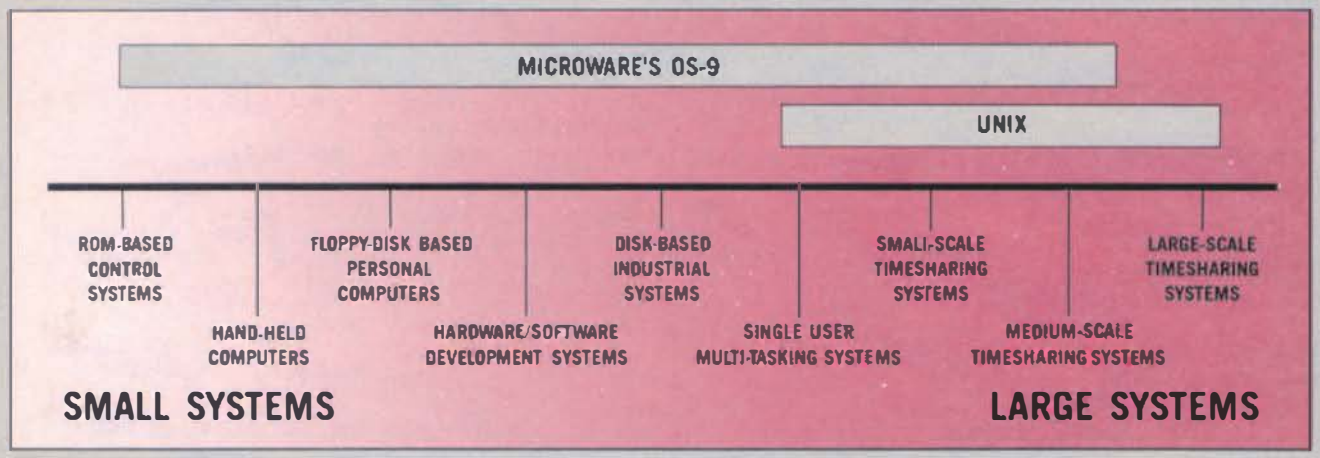
Up to Three Terminals - Instant expansion

*Trademark of Technical Systems Consultants

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Only Microware's OS-9 Operating System Covers the Entire 68000 Spectrum



Is complicated software and expensive hardware keeping you back from Unix? Look into OS-9, the operating system from Microware that gives 68000 systems a Unix-style environment with much less overhead and complexity.

OS-9 is versatile, inexpensive, and delivers outstanding performance on any size system. The OS-9 executive is much smaller and far more efficient than Unix because it's written in fast, compact assembly language, making it ideal for critical real-time applications. OS-9 can run on a broad range of 8 to 32 bit systems based on the 68000 or 6809 family MPUs from ROM-based industrial controllers up to large multiuser systems.

OS-9'S OUTSTANDING C COMPILER IS YOUR BRIDGE TO UNIX

Microware's C compiler technology is another OS-9 advantage. The compiler produces extremely fast, compact, and ROMable code. You can easily develop and port system or application software back and forth to standard Unix systems. Cross-compiler versions for VAX and PDP-11 make coordinated Unix/OS-9 software development a pleasure.

SUPPORT FOR MODULAR SOFTWARE — AN OS-9 EXCLUSIVE

Comprehensive support for modular software puts OS-9 a generation ahead of other operating systems. It multiplies programmer productivity and memory efficiency. Application software can be built from individually testable software modules including standard "library" modules. The modular structure lets you customize and reconfigure OS-9 for specific hardware easily and quickly.

A SYSTEM WITH A PROVEN TRACK RECORD

Once an unde round classic, OS-9 is now a solid hit. Since 1980 OS-9 has been ported to over a hundred 6809 and 68000 systems under license to some of the biggest names in the business. OS-9 has been imbedded in numerous consumer, industrial, and OEM products, and is supported by many independent software suppliers.

Key OS-9 Features At A Glance

- Compact (16K) ROMable executive written in assembly language
- User "shell" and complete utility set written in C
- C-source code level compatibility with Unix
- Full Multitasking/multiuser capabilities
- Modular design - extremely easy to adapt, modify, or expand
- Unix-type tree structured file system
- Rugged "crash-proof" file structure with record locking
- Works well with floppy disk or ROM-based systems
- Uses hardware or software memory management
- High performance C, Pascal, Basic and Cobol compilers

microware® OS-9™

MICROWARE SYSTEMS CORPORATION
1866 NW 114th Street
Des Moines, Iowa 50322
Phone 515-224-1929
Telex 910-520-2535

Microware Japan, Ltd
3-8-9 Baraki, Ichikawa City
Chiba 272-01, Japan
Phone 0473(28)4493
Telex 299-3122

'68' MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
SO9-5/8 DMF disk-CDS1-8212W-Sprint 3 Printer

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

CONTENTS

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
Ph (615)842-4600 TELEX 558 414 PVT BTH

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES
USA
1-Year $24.50 2-Years $42.50 3-Years $64.50
FOREIGN

See Page 60

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands **ONLY** should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

All systems feature the **GIMIX CLASSY CHASSIS**; with a ferro-resonant constant voltage power supply, gold plated bus connectors, and plenty of capacity for future expansion.

Static **RAM** and double-density **DMA** floppy disk controllers are used exclusively in all systems.

All systems are guaranteed for 2 MHz operation and include complete hardware and software documentation, necessary cables, filler plates, etc.

Systems are assembled using burned-in and tested boards, and all disk drives are tested and aligned by **GIMIX**.

You can add additional components to any system when ordering, or expand it in the future by adding **RAM**, **I/O**, etc.

GIMIX lets you choose from a wide variety of options to customize your system to your needs.

SOFTWARE

All **OS-9/FLEX** systems allow you to software select either operating system.

Also included is the **GMXBUG** monitor and, in systems with 128K or more of RAM, **GMX-VDISK** for **FLEX**.

All **GIMIX OS-9** systems include **Microware's Editor, Assembler, Debugger, Basic09,** and **Runb**; and the GMX versions of **RMS** and **DO** for **OS-9**.

All **GIMIX** versions of **OS-9** can read and write RS color computer format OS-9 disks, as well as the Microware/GIMIX standard format.

New and exclusive with **OS-9 GMX III** systems is the **GMX OS-9 Support ROM**, a monitor for **OS-9** that includes memory diagnostics and allows the system to boot directly from either hard disk or floppy.

A wide variety of languages and other software is available for use with either **OS-9** or **FLEX**.

OS-9 GMX III/FLEX SYSTEMS (#79)

The **#79** super system now includes (in addition to the above): the **GMX 6809 CPU III**, a **256K CMOS Static RAM Board (#72)**, and a **3-port Intelligent Serial I/O Processor (#11)**.

The **GMX 6809 CPU III** can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	**$5998.79**
with dual 80 track DSDD drives	**$6198.79**
with **#88** dual 8" DSDD drive system	**$7698.79**
with **#90** 19MB Winchester subsystem and one 80 track	**$8898.79**
with a 47MB Winchester subsystem and one 80 track	**$10,898.79**
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	**$12,398.79**

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money. Or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

OS-9 GMX I / FLEX SYSTEMS #49

The **#49** systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$3998.49**
with dual 80 track DSDD drives	**$4198.49**
with **#88** dual 8" DSDD drive system	**$5698.49**
with **#90** 19MB Winchester subsystem and one 80 track	**$6898.49**

OS-9 GMX II / FLEX SYSTEMS #39

The **#39** systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$4498.39**
with dual 80 track DSDD drives	**$4698.39**
with **#88** dual 8" DSDD drive system	**$6198.39**
with **#90** 19MB Winchester subsystem and one 80 track	**$7398.39**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8" about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

Simply add additional memory and I/O boards. Your GIMIX system can grow with your needs. Contact us for a complete list of available boards and options.

#72 256KB CMOS STATIC RAM board with battery back up	**$1898.72**
#64 64KB CMOS STATIC RAM board with battery back up	**$528.64**
#67 64KB STATIC RAM board	**$478.67**
#11 3 port intelligent serial I/O board	**$498.11**
#43 2 port serial I/O board	**$128.43**
#42 2 port parallel I/O board	**$88.42**
#95 cable sets (1 needed per port), specify board	**$24.95**

NOW SHIPPING !

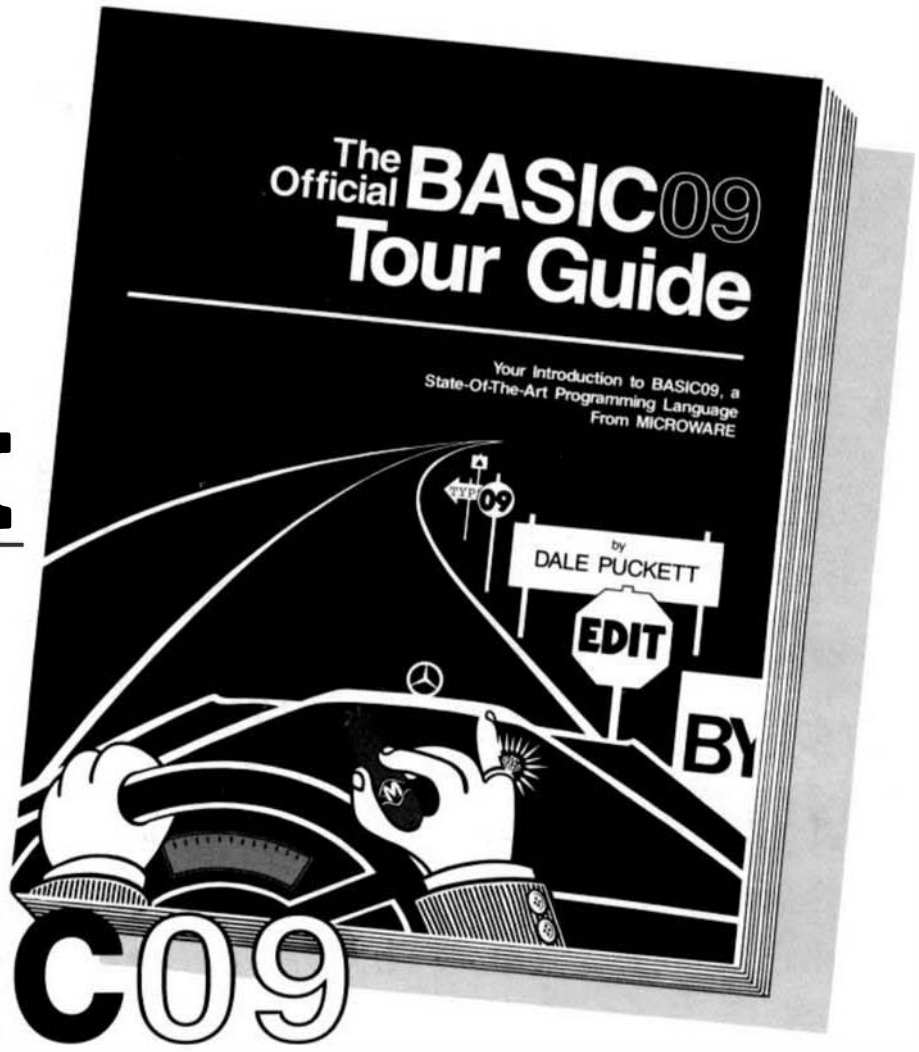
UniFLEX

GMX III Systems

GIMIX INC.

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

Get the most out of BASIC09



The **OFFICIAL BASIC09 TOUR GUIDE** is skillfully written in a friendly and easy-to-read style. Just perfect for those new to computers and to BASIC09. It's also a *valuable reference* book for programmers, engineers, students and hobbyists, providing an in-depth look at BASIC09 plus an overview of the OS-9 operating system. Comprehensive reference sections on BASIC09 and OS-9 commands are also included.

The book "maps" out your route through the Mercedes of Basics . . . BASIC09 and puts you in the driver's seat in no time. Fasten your seatbelt, sit back and enjoy the ride to perfecting your programming skills.

MICROWARE . . .

The **OFFICIAL BASIC09 TOUR GUIDE** comes from the people who wrote BASIC09. As the leader in 6809 system software, we at MICROWARE care about our users and want to help you get the most from our products.

It's Easy to Order.

Phone orders are accepted from MasterCard or VISA cardholders or for COD shipment. You can also order by mail using the coupon below. Quantity discounts are available to educational organizations and dealers. For further information contact Microware.

microware®

Specialists in system software for 68-family microprocessors since 1977.

OS-9 and BASIC09 are trademarks of Microware and Motorola.

Microware Systems Corporation
1866 N.W. 114th Street
Des Moines, Iowa 50322
Telephone 515/224-1929
Telex 910-520-2535

Please send _____ copies of the **Basic09 Tour Guide** book at $18.95 each. Add $2.00 for UPS shipping in the U.S. or $5.00 for overseas air mail per book. Iowa residents add 4% sales tax.

Name _____

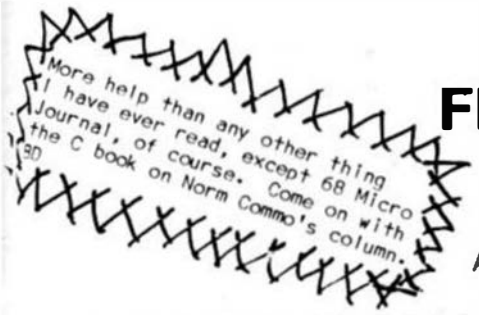
Address _____

City _____ _____

State _____ Zip _____

☐ I have enclosed a check

☐ Charge to my bank card:
MasterCard ☐ VISA ☐
Card Number _____
Expiration _____



FLEX™ USER NOTES
THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1,.C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

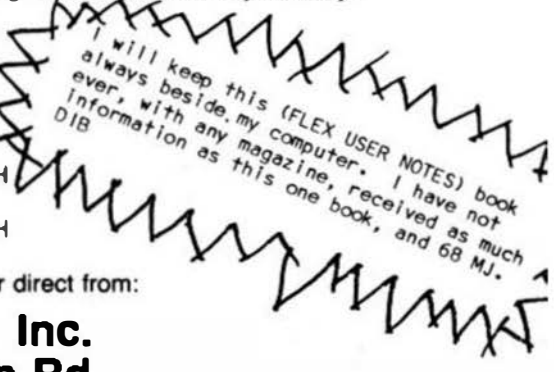
Foreign Orders Add $4.50 S/H

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H



See your local S50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

TELEX 558 414 PVT BTH

™FLEX is a trademark of Technical Systems Consultants

THE 68000 FROM SMOKE SIGNAL!

ADD 68000 AND UNIX™ * TO YOUR EXISTING SS-50 COMPUTER AT PRICES 50% TO 75% OFF LIST

THANK YOU

Seven years ago, Smoke Signal was founded to sell state-of-the-art computer products, by mail, to individual professional programmers and hardware engineers. At that time, most big companies did not believe in the power or future of microcomputers for serious computing applications. Only after you, the individual computer user, proved the viability of the microcomputer was Smoke Signal able to sell systems for business uses. However, as we progressed to become the leader in SS-50 systems, we had to add the sales and technical support services demanded by these business customers — and our prices for complete systems reflected these added costs.

With the introduction of our 68000 products, we wanted to find a way to say thanks to you, our original customers, the individual computer users, and still offer complete sales and technical support to our business customers for complete systems. We think this offer accomplishes both of these goals. We are offering you a choice of upgrade kits that will bring any SS-50 computer up to the electrical equivalent of our complete 68000 computer systems at prices far below complete system prices. In fact, the prices offered are 50% or more off our normally low prices for the components contained in the upgrade kits.

This special offer is limited to one upgrade kit per customer and is our way of saying thanks to those of you who had confidence in us from the beginning.

THE UPGRADES

The following upgrade kits were designed so that any SS-50 system can be upgraded to 68000/UNIX.

SWTP UPGRADE..... $2,800.00
Contains: LMB-1A SS-50C Motherboard, DCB-4A floppy controller, PSA-1 Winchester/Tape DMA interface, SCB-68K 68000 CPU, SER-2 dual serial board, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

GIMIX UPGRADE..... $2,500.00
Contains: Same as SWTP Upgrade except allows you to use your GIMIX motherboard, serial board and Winchester power supply.

Users of standard SMOKE SIGNAL systems may choose one of the following upgrade kits:

For SSB floppy based systems:

SS-FD UPGRADE..... $2,100.00
Contains: SCB-68K 68008 CPU, PSA-1 Winchester/Tape DMA interface, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

For SSB Winchester based systems:

SS-HD UPGRADE..... $500.00
Contains: SCB-68K 68008 CPU and REGULUS.

COMPLETE SYSTEMS

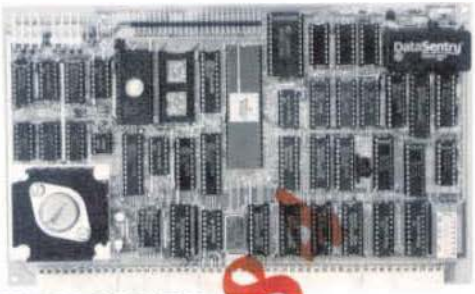
SMOKE SIGNAL is also making available complete VAR/68K™ systems at dramatic discounts. This offer is only available through SMOKE SIGNAL dealers. Contact SMOKE SIGNAL directly for information about how to order a complete VAR/68K system.

RULES OF THE OFFER

1) Limit, one upgrade system per customer.
2) Prices valid through December 31, 1984.
3) Orders must be accompanied by full payment in the form of individual check or credit card authorization.
4) Support will only be provided for systems containing the following SMOKE SIGNAL boards: SCB-68K, DCB-4A, PSA-1, and a motherboard such as the LMB-1A with extended addressing and main terminal I/O at FF7E8.
5) While we feel that most static RAM boards will work with these upgrades, we only guaranty compatibility with systems containing SMOKE SIGNAL static or dynamic RAM.

VAR/68K is a trademark of Smoke Signal.
REGULUS is a registered trademark of Alcyon Corp.; UNIX is a registered trademark of Bell Laboratories; OS9 and OS9/68K are trademarks of Microwar ; MACSBUG is a trademark of Motorola Inc.

*Regulus the OS offered is UNIX Compatible



PRODUCTS

The heart of all these upgrade kits is SMOKE SIGNAL'S new SCB-68K 8 MHz 68008 CPU Board. This standard (5½" x 8") board will replace a SCB-69 CPU Board in any SMOKE SIGNAL computer with current revision boards. This board contains a real-time clock with battery back-up, 2 EPROM slots for up to 64K bytes of storage, a MACSBUG™ type monitor along with an auto boot loader and a mnemonic disassembler, plus many more features.

All upgrades also come standard with REGULUS™, a UNIX like operating system which is totally compatible with UNIX. REGULUS supports real-time tasks, shared memory, record locking and contains a shell similar to the Berkeley C shell. Along with the operating system, you get C, an editor, assembler, linking loader, interactive debugger and a word processor.

SMOKE SIGNAL is also including in many of the kits the DCB-4A double density floppy controller which can handle up to four 5" and four 8" floppies and contains 1K of buffer RAM for fast disk transfers; the PSA-1 Winchester/Tape DMA interface board which has taps for SASI and Priam disk interfaces as well as a tap for 90 ips tape streamers which are supported under both REGULUS and OS9™; either a M-256-X or M-512-X dynamic RAM board with over two years of field proven reliability; and the LMB-1A heavy duty motherboard with gold plated connectors, extended addressing and on-board baud rate generator with ten selectable baud rates.

SOFTWARE

Software and Software Support is available only from Smoke Signal dealers. Spread Sheet, Word-Processing, Relational Database, C, Basic and Cobol are all available now. Additional system's software is becoming available every day because of the UNIX compatibility.

SMOKE SIGNAL dealers are also offering Microware's OS9/68K™ to purchasers of these upgrade kits. SMOKE SIGNAL will offer other Microware 68000 products as they become available.

SUPPORT

Even at these "lower than PC" prices, we're not going to leave you with "PC" type support. We've arranged with one of our very technically qualified dealers to provide you with add-on software and technical support. In addition to answering your questions on how to convert your system to the 68000, he has a group of his customers who are themselves computer experts who are joining in a network that will help with even the most technical questions. We hope you will contribute your ideas to the network so that we all can benefit from new and fresh thinking. Complete details of the support available are included with the upgrade systems.

ORDER FORM

Fill in your name, address and phone number below. Your order will be shipped UPS so please do not use P.O. Box. Check items being ordered on form. Add prices for all items selected. CA residents must add 6% for sales tax. Total the amount for your order and check payment method below.

Name _____
Address _____
City, State, Zip _____
Phone _____

Payment: ☐ Enclosed Check
☐ VISA
Mastercard
Card # _____
Exp. Date _____
Signature _____

☐ SS-FD UPGRADE	$2100	_____
☐ SS-HD UPGRADE	500	_____
☐ SWTP UPGRADE	2800	_____
☐ GIMIX UPGRADE	2500	_____
☐ M-256X RAM	648	_____
☐ M-512-X RAM	948	_____
☐ SER-2 I/O	85	_____
☐ 20Mb HARD DISK (instead of 5Mb)	800	_____
Sub Total		_____
CA residents add 6%		_____
Total		_____

SEND COMPLETED ORDER FORM TO:
SMOKE SIGNAL
31336 Via Colinas, Westlake Village, CA
91362-3964
(818) 889-9340

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

Change of Pace

This time, I thought I would do something I have not done for some time. I've been working on some software to allow two computers in the same room to communicate with each other. At home, I have my old SWTPc frame sitting 6 feet away from a Peripheral Technology PT69 system. At work, I have several systems in close proximity. It seemed to me that we could easily adapt some modem software to do the job. Having written some software for modem use, I tried it out as an intercomputer communications link via RS 232 serial ports and direct connections. Several problems became apparent very quickly. The modem software displays the text being transmitted, on the terminal at both the sending and receiving end. This limits the baud rate on the communication line to 2400 or so with a terminal running at 9600. Secondly the system can only transmit and receive text or ASIII files. Binary files can contain any of the possible 256 combinations of 8 bits, and so there can be no unique control codes to allow handshaking. My original modem program simply fills memory with incoming text until the user types an ESCAPE, at which time the memory contents are saved to a disk file. If the file is longer than the memory buffer, overflow occurs and the remainder of the file is lost.

In an effort to solve some of these problems I wrote a couple of utilities in PL/9 called BINASC and ASCBIN. The first converts a binary file, read from a disk, character by character to ASCII, similar to what a memory dump utility would do. That is it converts $B6 into two ASCII characters, 'B and '6 the codes for which are $42 and $36. The digits 0-9 and letters A-F are of course within the range of "printable" ASCII codes, and the file is converted to characters within this range. Control codes can now be used for handshaking in the system.

With these utilities, I would convert a binary file to ASCII form and save it as a file, use my MODEM program to transmit it to the receiving end, save it as an ASCII file, and run ASCBIN to convert it back to binary form. That worked fine but was cumbersome and still doesn't solve the problem of memory overflow. I decided to write a pair of utilities called TSEND and T EC, to send and receive text files. They were written so that the sending end would send 10000 characters and then a "waiting" control code. The receiving end interprets WAITING as a command to save the buffer contents to a disk and send back a "START" code when ready for more. By this means, a very long file may be transmitted. In writing these utilities, I removed the feature of displaying all the text on the terminal, and made the SEND program display an asterisk (*) on the screen for each 256 bytes transmitted. I felt that the user ought to see something happening.

The receive program prompts for deletion of the file if it already exists, and indicates that the program is ready to receive a file. While stopped to write to the disk, the terminal displays "WRITING TO DISK", and when finished, the terminal displays "DONE". This is all nice, but now I had four utilities, two to convert files, and two to send and receive them.

The next step of course was the thought that I could build the BINASC and ASCBIN into the SEND and RECEIVE programs respectively and have what I called BSEND and BREC for transmission of binary files. Now I had eliminated BINASC and ASCBIN as separate utilities, but I had two pair of utilities for sending TEXT and BINARY files respectively. Now what would the next step be? Let the SEND program read the first byte of the file. If it is $02, the file is binary. Set the BINARY flag in the file and signal the receiver to do the same. Now add some logic to decide, based on the file type, whether or not to do the BINASC and ASCBIN routines, and you have a single pair of utilities to send and receive any type of file, of any length between two computers running FLEX. I've tried running them at 9600 baud, and I find that the transmission rate is just faster than 1 sector per second for a text file, and just slower than 1 sector per second for a binary file.

Since there is no critical timing in the program, I expect that it will allow transmission at 19.2K baud and perhaps 38.4 K baud as well. I assumed that it should be possible to send signals reliably over 10 feet of wire with RS-232 signal levels. The software has no CRC or parity checking included for this reason. I've sent a whole 130 sector binary file back and forth several times at 9600 baud with no errors. This pair of utilities will greatly facilitate moving files containing software developed for customer's machines between our various systems. We will be able to transfer the program file from the R&D department to the Production department for the programming of EPROMS with little effort.

I offer these utilities to show that it is possible to write concise code that compiles efficiently in PL/9. USEND compiles to 688 bytes and UREC to about 940. The utilities occupy 3 and 4 sectors respectively. It would be possible to put USEND in the utility command space at $C100, and the same could be done with UREC, but it wouldn't make much sense to do so since a 10000 byte buffer is required for the program to operate, so like the TSC COPY and FORMAT utilities, it must occupy some low memory anyway.

I think these utilities indicate how well PL/9 may be used to write "system software". I use it every day for logic and control applications that do considerable calculations including scientific functions. Of course with its 6+ digit floating point arithmetic, PL/9 is not intended for use in financial calculations. It is not suited for mortgage payment schedules or interest on the national debt, but in many applications it really works very very well. So much for the sales pitch on PL/9. I really don't mean it that way, as I have nothing to gain by the sales of PL/9 except to see the folks who supply it reap some of the rewards they deserve for their efforts in creating it.

Actually, my "change of pace" is not entirely that. I recently received a letter from Graham Trott, the author of PL/9 and it contains a very quotable passage regarding the Compiler Assembler debate. I will quote it here.

"While I'm putting finger to key, I'd like to comment on the continuing compiler vs assembler debate. I agree that to put together standard library routines in assembler is much like using a primitive compiler, with the disadvantage that the result is rarely as readable as the equivalent high-level program, nor frequently any more efficient. I will make a further observation, however. PL/9 itself is written in assembly language. Whatever the merits of the code it produces, I believe that no compiler written in a high-level language could approach PL/9 in the number of facilities resident TOGETHER in 64K of memory. Only assembly language can offer such compactness. Consider two points, however. Firstly, compactness only matters if we have limited memory (which many modern systems do not). Secondly, do the rewards justify the effort needed to create 16K of tightly-written code? I could have written two or three compilers in 'C' in the time it took to write and debug PL/9, and although the result may not have been so useful to the customers, they would probably have been more profitable to me. I'm not being cynical, just trying to look at the prob m from a commercial point of view. If you're willing (and able) to spend the time and effort you'll probably end up with a better product if you write it in assembly language, but I doubt if you'll be able to justify it in advance on cost grounds. My advice is: Write it in a H.L, then see if you need to make it faster or smaller. If not, don't waste time on it."

```
/* PROGRAM TO SEND A TEXT OR BINARY FILE VIA WIRE AT 4800 BAUD.
   BINARY FILE IS CONVERTED TO ASCII FOR TRANSMISSION.
   BY INITIALIZING THE ACIA TO DIVIDE BY 64, THIS CAN
   ALSO BE USED FOR A 1200 BAUD MODEM WITHOUT RECONFIGURING
   THE PORT HARDWARE      */

CONSTANT
   BIN = 0,
   TXT = 1,
   END = 9,
   START = 6,
   WAITING = 7;

AT $E000 BYTE MODEM(2);
AT $E004 BYTE TERM(2);

INCLUDE TRUFALSE.DEF.1;
INCLUDE FLEX.LIB.1;


PROCEDURE PUTCHAR(BYTE .DEVICE: BYTE CHAR);
```

```
   REPEAT UNTIL DEVICE(0) AND 2
   DEVICE(1)=CHAR;
ENDPROC;


PROCEDURE GETCHAR (BYTE .DEVICE);
   REPEAT UNTIL DEVICE(0) AND 1;
ENDPROC BYTE DEVICE(1);


/* GETIF RETURNS CHAR IF ONE HAS BEEN INPUT, ELSE RETURNS NULL.
   THIS PROCEDURE DOES NOT WAIT FOR A CHARACTER  */


PROCEDURE GETIF (BYTE .DEVICE);
   IF DEVICE(0) AND 1 THEN RETURN DEVICE(1);
ENDPROC BYTE 0;


PROCEDURE BINASC(BYTE CHAR, .UNIBBLE, .LNIBBLE);
   LNIBBLE = CHAR AND $0F;
   UNIBBLE = SHIFT(CHAR,-4) AND $0F;
   IF LNIBBLE < 10 THEN LNIBBLE = LNIBBLE + $30
      ELSE LNIBBLE = LNIBBLE + $37;
   IF UNIBBLE < 10 THEN UNIBBLE = UNIBBLE + $30
      ELSE UNIBBLE = UNIBBLE + $37;
ENDPROC;

PROCEDURE SEND_LOCAL : BYTE CH, CH1, CH2, BIN_FILE, INFILE(320):
                       INTEGER COUNT;

   GET FILENAME(.INFILE);
   SET_EXTENSION(.INFILE,TXT); /* DEFAULT TEXT FILE */
   MODEM(0)=3;
   MODEM(0)=$15; /* INITIALIZE DIVIDE BY 16 CLOCK */
   OPEN FOR READ (.INFILE);
   IF INFILE(1)<>0 THEN
   BEGIN
      REPORT_ERROR(.INFILE);
      FLEX;
   END;
   CH = READ(.INFILE);
   IF CH = $02 THEN
   BEGIN
      SET BINARY(.INFILE);
      BIN_FILE = TRUE;
      PUTCHAR(.MODEM,BIN);
   END
   ELSE
   BEGIN
      BIN FILE = FALSE;
      PUTCHAR(.MODEM,TXT);
   END;
   CH1= GETCHAR(.MODEM); /* HAS TO BE START CHAR */
   IF BIN FILE THEN BINASC(CH, .CH1, .CH2);
   COUNT := 0;

   WHILE INFILE(1)=0
   BEGIN
      REPEAT
         IF BIN_FILE THEN
         BEGIN
            PUTCHAR(.MODEM,CH2);
            COUNT = COUNT+2;
         END
         ELSE
         BEGIN
            PUTCHAR(.MODEM,CH);
            COUNT = COUNT + 1;
         END;
         IF COUNT AND $00FF = 0 THEN PUTCHAR(.TERM,'*');
         CH = READ(.INFILE);
         IF BIN FILE
            THEN BINASC(CH, .CH1, .CH2);
      UNTIL COUNT = 10000 .OR INFILE(1)<>0
      IF INFILE(1)=0 THEN
      BEGIN
         PUTCHAR(.MODEM,WAIT)=0;
         COUNT = 0;
         REPEAT UNTIL GETIF(.MODEM)= START;
      END;
   END;
   IF INFILE(1)<>0 THEN REPORT_ERROR(.INFILE);
   CLOSE_FILE (.INFILE);
   PUTCHAR(.MODEM,END);


/* PROGRAM TO RECEIVE A TEXT OR BINARY FILE VIA WIRE AT 4800 BAUD.
   BINARY FILE HAS BEEN CONVERTED TO ASCII BY SEND PROGRAM, AND
   THIS PROGRAM CONVERTS IT BACK TO BINARY FORM.
   BY INITIALIZING THE ACIA TO DIVIDE BY 64, THIS CAN
   ALSO BE USED FOR A 1200 BAUD MODEM WITHOUT RECONFIGURING
   THE PORT HARDWARE.      */

ORIGIN = 0;
STACK = $D0FF;

CONSTANT
   BIN     = 0,
   TXT     = 1,
   END     = 5,
   START   = 6,
   WAITING = 7;

AT $1000 BYTE DATA(10010);
AT $E000 BYTE MODEM(2); /* 6850 MODEM PORT ADDRESS */
AT $E004 BYTE TERM(2);  /* 6850 TERMINAL PORT ADDRESS */
```

```
INCLUDE TRUFALSE.DEF.1;
INCLUDE FLEX.LIB.);  /* FLEX FILE HANDLING INTERFACE */


PROCEDURE PUTCHAR(BYTE .DEVICE: BYTE CHAR);
   REPEAT UNTIL DEVICE(0) AND 2;
   DEVICE(1)=CHAR;
ENDPROC;


PROCEDURE GETCHAR (BYTE .DEVICE);
   REPEAT UNTIL DEVICE(0) AND 1;
ENDPROC BYTE DEVICE(1);


PROCEDURE PRINT(BYTE .STRING):BYTE N;
   N=0;
   WHILE STRING(N)
   BEGIN
      PUTCHAR(.TERM,STRING(N));
      N=N+1;
   END;
ENDPROC;


PROCEDURE CRLF;
   PUTCHAR(.TERM,$0D);
   PUTCHAR(.TERM,$0A);
ENDPROC;


PROCEDURE ASCBIN(BYTE UNIBBLE, LNIBBLE);
   IF UNIBBLE >$40 THEN UNIBBLE = UNIBBLE - $37
      ELSE UNIBBLE = UNIBBLE - $30;
   IF LNIBBLE >$40 THEN LNIBBLE = LNIBBLE - $37
      ELSE LNIBBLE = LNIBBLE - $30;
ENDPROC BYTE(SHIFT(UNIBBLE,4) + LNIBBLE);


/* MAIN PROGRAM STARTS HERE */


PROCEDURE REC_LOCAL : BYTE CH, CH1, BIN_FILE, OUTFILE(320):
                      INTEGER INDEX,LIMIT,DBYTE;
   CRLF;
   GET FILENAME(.OUTFILE);
   SET_EXTENSION(.OUTFILE,TXT);
   MODEM(0)=3;
   MODEM(0)=$15; /* INITIALIZE DIVIDE BY 16 CLOCK */
   OPEN_FOR_WRITE (.OUTFILE);

   IF OUTFILE(1)=3
      THEN BEGIN
             PRINT "DELETE EXISTING FILE? ";
             CH = GETCHAR(.TERM);
             PUTCHAR(.TERM,CH);
             CRLF;
             IF CH = 'Y .OR CH = 'y
               THEN BEGIN
                      DELETE FILE(.OUTFILE);
                      OPEN FOR WRITE(.OUTFILE);
                    END ELSE FLEX;
             END;
   IF OUTFILE(1)<>0 THEN REPORT_ERROR(.OUTFILE);
   PRINT "READY TO ACCEPT A FILE"; CRLF;
   CH = GETCHAR(.MODEM);
   IF CH = BIN THEN
   BEGIN
      SET_BINARY(.OUTFILE);
      BIN_FILE = TRUE;
   END
   ELSE BIN_FILE = FALSE;
   INDEX = 0;
   LIMIT = 0;
   PUTCHAR(.MODEM,START);
   REPEAT
      CH = GETCHAR(.MODEM); /* WAIT FOR A CHARACTER */
      CH1 = CH; /* SAVE IT */
      IF CH <> END .AND CH <> WAITING
         THEN
         BEGIN
            DATA(INDEX) = CH;
            INDEX = INDEX+1;
         END
         ELSE LIMIT = INDEX; /* ONE PAST LAST VALID CHAR */
      IF LIMIT<>0 THEN
      BEGIN
         PRINT "  WRITING TO DISK";
         PUTCHAR(.TERM,$001);
         INDEX=0;
         WHILE INDEX < LIMIT
         BEGIN
            IF BIN_FILE THEN
            BE IN
               CH = ASCBIN(DATA(INDEX),DATA(INDEX+1));
               INDEX = INDEX+2;
            END
            ELSE
            BEGIN
               CH = DATA(INDEX);
               INDEX = INDEX+1;
            END;
            WRITE(.OUTFILE, CH);
         END;
         PRINT"                    ";
         PUTCHAR(.TERM,$0D);
```

```
        PUTCHAR(.MODEM.START);
        LIMIT=0;
        INDEX=0;
      END;
  UNTIL CH1 = END;
  CLOSE FILE1.OUTFILE1; CRLF;
  PRINT "DONE";
  CRLF;


/*  TRUE - FALSE - NEW   DEFINITIONS  V:3.01  */

/*
  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  *  THE FOLLOWING HAVE BEEN REMOVED FROM THE LANGUAGE AS A PART  *
  *  OF THE UPGRADE BETWEEN VERSION 2.XX AND VERSION 3.XX         *
  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*/
  CONSTANT TRUE=-1, FALSE=0;

  AT $0000:BYTE NEW;


    /*  FLEX INTERFACE ROUTINES  */

ASMPROC FLEX;
  GEN $7E,$CD,$03;                    /* JMP $CD03 */

ASMPROC GET FILENAME(INTEGER);
  GEN $AE,$62;                        /* LDX 2,S */
  GEN $BD,$CD,$20;                    /* JSR $CD20 */
  GEN $25,$02;                        /* BCS *+4 */
  GEN $6F,$01;                        /* CLR 1,X */
  GEN $39;                            /* RTS */

ASMPROC SET EXTENSION(INTEGER,BYTE);
  GEN $AE,$63;                        /* LDX 3,S */
  GEN $A6,$62;                        /* LDA 2,S */
  GEN $BD,$CD,$33;                    /* JSR $CD33 */
  GEN $39;                            /* RTS */

ASMPROC REPORT ERROR(INTEGER);
  GEN $AE,$62;                        /* LDX 2,S */
  GEN $7E,$CD,$3F;                    /* JMP $CD3F */

ASMPROC OPEN FOR READ(INTEGER);
  GEN $AE,$62;                        /* LDX 2,S */
  GEN $86,$01;                        /* LDA #1 */
  GEN $A7,$84;                        /* STA D,X */
  GEN $7E,$D4,$06;                    /* JMP $D406 */

ASMPROC READ(INTEGER): BYTE;
  GEN $34,$10;                        /* PSHS X */
  GEN $AE,$64;                        /* LDX 4,S */
  GEN $BD,$D4,$06;                    /* JSR $D406 */
  GEN $1F,$89;                        /* TFR A,B */
  GEN $35,$90;                        /* PULS X,PC */

ASMPROC OPEN FOR WRITE(INTEGER);
  GEN $AE,$62;                        /* LDX 2,S */
  GEN $86,$02;                        /* LDA #2 */
  GEN $A7,$84;                        /* STA 0,X */
  GEN $7E,$D4,$06;                    /* JMP $D406 */

ASMPROC WRITE(INTEGER,BYTE);
  GEN $AE,$63;                        /* LDX 3,S */
  GEN $A6,$62;                        /* LDB 2,S */
  GEN $7E,$D4,$06;                    /* JMP $D406 */

ASMPROC SET BINARY(INTEGER);
  GEN $AE,$62;                        /* LDX 2,S */
  GEN $86,$FF;                        /* L A #$FF */
  GEN $A7,$88,$38;                    /* STA 59,X */
  GEN $39;                            /* RTS */

ASMPROC CLOSE FILE(INTEGER);
  GEN $AE,$62;                        /* LDX 2,S */
  GEN $86,$04;                        /* LDA #4 */
  GEN $A7,$84;                        /* STA 0,X */
  GEN $7E,$D4,$06;                    /* JMP $D406 */

ASMPROC DELETE FILE(INTEGER);
  GEN $AE,$62;                        /* LDX 2,S */
  GEN $86,$0C;                        /* LDA #12 */
  GEN $A7,$84;                        /* STA 0,X */
  GEN $E6,$04;                        /* LDB 4,X */
  GEN $BD,$D4,$06;                    /* JSR $D406 */
  GEN $E7,$04;                        /* STB 4,X */
  GEN $39;                            /* RTS */
```


"C" User Notes

Edgar M. (Bud) Pass, Ph.D.
1454 Latta Lane
Conyers, GA 30207

INTRODUCTION

This month's column provides some follow-up information to last month's column on problems in the 6809 Full C compilers and begins a discussion of string-handling in the C language.

FOLLOW-UP

Since the last article was prepared, I have received several letters providing some feedback to some of the points raised in the column. I also received a new version of Windrush C (McCosh C for FLEX).

Edward Dunham reported that the newest version of Introl C (v1.5.4) correctly processed an example taken from the "C Puzzle Book", which v1.4 had processed correctly, but which earlier versions of v1.5 had processed incorrectly.

The new version of Windruah C solved the simple string initialization problem which caused the compiler to abort with an undocumented error message.

James McCosh, the developer of the Microware, SWTPC, and Windrush C compilers, responded to the issues raised in the earlier article.

He promised to modify the "toupper" and "tolower" macros to check their arguments, as the current version of UNIX C does. This is one of the more inconsistent aspects of the low-level C libraries across the different C implementations. It may be traced back to K & R, which does not specify the argument-checking.

He acknowledged the problems with structure initialization, and promised that it will be corrected in a future version. To avoid this problem, a structure which logically contains a character string should rather contain a a pointer to a character string. For example, use

```
structure x{char *c;} y = {"abcd"};
```

rather than

```
structure x{char c[5];} y = {"abcd"};
```

He stood firm on the twenty-level dereferencing problem (••••••••••••••••••••x) diacuaaed in the earlier article. His compilers handle five levels of dereferencing properly, and he considers this sufficient. Since this ia an exceptional, contrived example, I tend to agree with him that this is not a major problem. In addition, I have determined that many of the UNIX and UNIX-clone C compilers also do not correctly process the example correctly. However, I suggested that he document the limitation in the manuals, rather than letting the user discover the situation accidentally, or somehow design a data structure which depends upon more than five levels of indirect referencing.

He reinforced my experiences in attempting to port software across versions of C compilers, whether on the same or on different operating systems and processors. He found that the IBM PC Microsoft Lattice C compiler handled structures differently from his compilers and attempted to optimize the produced code by incorrectly maintaining commonly-used values in registers. He noted that he produced the 68000 version of his compiler in about three weeks and regrets not having produced an 8086 version of it, rather that attempting to port C programs to another compiler.

STRING-HANDLING IN C

One of the most limiting aspects of the C language

ia in its handling of character strings. The language has no direct counterpart of the BASIC string-handling syntax, such as the "+" operator for concatenation, the "MID$", "LEFT$", "RIGHT$" functions, string assignment and comparison, nor does it have such facilities as automatic allocation and deallocation of string space nor arbitrary contents.

The only manner in which to provide string-handling facilities in the C language such as those comparable to BASIC string-handling facilities is through the definition of functions.

The C language provides only one form of string constant: fixed-length, null-terminated, and enclosed in double-quote symbols. It provides a notation for single-character constants enclosed in apostrophes, which are not strings, but a representation of a value between 0 and 255, 0 and 127, -128 to +127, or whatever the C implementation defines it to be.

Because of the limitations of the C language, many C programs process strings one character at a time, rather than as logical groups. This is unfortunate, as it detracts from the efficiency of the generated code and also makes the C program harder to read, to debug, to maintain, and to port to another C compiler. Although the string-handling functions may themselves process the strings one character at a time, the calls to the functions will be integral and will indicate the concepts, rather than the details.

What is needed is the definition of a consistent set of string-processing functions, which may be used in programs, as required. Because of space, efficiency, and other considerations, the definition of several groups of functions is appropriate. Strings may be of fixed or variable length, restricted or arbitrary contents, etc. The more complex cases will involve variable-length strings with arbitrary contents requiring automatic allocation and deallocation. If this generality is not required in a given program, there should be available alternative simpler groups.

Although it will not provide a comprehensive solution, the discussion which appears below will start the definition of a string-handling library including groups of functions of differing capabilities. It is useful also for educational purposes to show how a knowledgeable C programmer breaks the overall problem into manageable functions and actually accomplishes the string manipulations. In a production environment, some of the C functions would almost certainly be recoded into assembly language for reasons of efficiency.

The first group of functions is due to Richard O'Keefe, who is concerned with precise definition and development of a library of string-processing functions. The names and usage of all of his functions will be provided below and in subsequent articles, but the detailed definitions will be provided only for the shorter and more interesting functions. The library itself will be available on the C Node being set up by the Atlanta Computer Society and from myself.

O'Keefe's library is divided into four primary groups, as follows:

str<oper> = null-terminated
strn<oper> = length-/null-terminated
mem<oper> = length-terminated
b<oper> = length-terminated

in which <oper> represents one of a large number of string operations.

He also provides header files and macros which support the functions themselves and the usage of the functions.

Following is a summary of the functions provided by O'Keefe's library. Note that all assume non-automatic storage allocation and deallocation. Thus, the user is responsible for actually ensuring that the string space is established and is sufficiently large for each string and for each string operation.

bcmp(s1, s2, len) returns 0 if the "len" bytes starting at "s1" are identical to the "len" bytes starting at "s2", non-zero if they are different.
bcopy(src, dst, len) copies "len" bytes from the source "src" to the destination "dst".
bfill(dst, len, fill) copies "len" fill characters to "dst".
bmove(dst, src, len) copies "len" bytes from the source "src" to the destination "dst".
bzero(dst, len) copies "len" 0x00 bytes to "dst".

bzero(dst, len) copies "len" 0x00 bytes to "dst".
int2str(dst, radix, val) converts the (long) integer "val" to character form and copies it to the destination string "dst" followed by a terminating null.
memccpy(dst, src, chr, len) copies bytes from "src" to "dst" until either "len" bytes have been moved or a byte equal to "chr" has been moved.
memchr(src, chr, len) searches the memory area pointed to by "src" extending for "len" bytes, looking for an occurrence of the byte "chr".
memcmp(lhs, rhs, len) compares the two memory areas "lhs[0..len-1]" and "rhs[0..len-1]".
memcpy(dst, src, len) copies "len" bytes from "src" to "dst" and returns a pointer to "dst".
memmov(dst, src, len) copies "len" bytes from "src" to "dst" and returns "dst"+"len".
memrchr(src, chr, len) searches the memory area pointed to by "src" extending for "len" bytes, looking for the last occurrence of the byte "chr".
memrev(dst, src, len) copies "len" bytes from "src" to "dst", in reverse order.
memset(dst, chr, len) fills the memory area "dst[0..len-1]" with "len" bytes all equal to "chr", and returns a pointer to "dst".
memtrans(dst, src, from, to, len) copies "len" characters from "src" to "dst", translating characters in "from" to corresponding characters in "to".
str2int(src, radix, lower, upper, &val) converts the string pointed to by "src" to an integer with radix "radix" and stores at "val".
strcat(s, t) concatenates "t" on the end of "s" and returns a pointer to "s".
strchr(s, c) returns a pointer to the first place in "s" where "c" occurs, or NULL if "c" does not occur in "s".
strcmp(s, t) returns > 0, = 0, or < 0 when "s" > "t", "s" = "t", or "s" < "t", according to the ASCII sequence of characters.
strcpy(dst, src) copies the characters starting with "src" to the area starting with "dst" until a null character is found, and returns a pointer to "dst".
strend(s) returns a character pointer to the null which ends "s".
strfind(src, pat) returns a pointer to the first occurrence of "pat" in "src", or returns NULL.
strkey(dst, head, tail, options) copies "tail"-"head" characters from "head" to "dst" according to the "options". It is intended to be used by the UNIX sort.
strlen(s) returns the number of characters in "s".
strmov(dst, src) copies the null-delimited string pointed to by "src" into "dst", and returns a pointer to the terminating null in "dst".
strncat(dst, src, n) copies up to "n" characters of "src" to the end of "dst".
strncmp(s, t, n) compares up to "n" characters of "s" and "t".
strncpy(dst, src, n) copies up to "n" characters from "src" to "dst".
strnend(src, len) returns a pointer to the end of the string pointed to by "src", of no more than "len" characters.
strnlen(src, len) returns the number of characters up to the first null in the string pointed to by "src", or "len", whichever is smaller.
strnmov(dst, src, n) copies up to "n" characters from "src" to "dst".
strnrev(dst, src, len) copies up to "len" characters from "src" to "dst" in reverse order.
strnrpt(dst, n, src, k) repeats the string "src" into "dst" "k" times, but truncates the result at "n" characters.
strntran(dst, src, len, from, to) copies up to "len" characters from "src" to "dst", translating characters in "from" to "to".
strrchr(s, c) returns a pointer to the last occurrence of "c" in "s", or NULL if "c" is not found in "s".

strrepl(dst, src, pat, rep, times) copies "src" to "dst", replacing the first "times" non-overlapping instances of "pat" by "rep".
strrev(dst, src) copies characters from "src" to "dst", in reverse order.
strrpt(dst, src, k) repeats string "src" into "dst" "k" times.
strtrans(dst, src, from, to) copies characters from "src" to "dst", translating characters in "from" to "to".
strsub(destination, source, offset, length) copies up to "length" bytes from "source"+"offset" to "destination".

Next month's column will provide the actual C text of of many of O'Keefe's string-processing functions, tailored to the McCosh and Introl C compilers for the 6809. His library attempts to be machine-independent, but requires the user to specify many parameters, such as the number of bits in an integer and whether chars are signed or unsigned.

C PROBLEM

This month starts a new feature of this column, in terms of an unsolved problem in C. The solutions will normally appear in the next month's column. As with the rest of the column, readers are invited to submit short C problems (hopefully illustrating some C concept or kink).

What is the problem with the following definition?:

```
#define tolower(x) (isupper(x) ? (x)|32 : (x))
```

Suggest at least two ways of solving the problem.

EXAMPLE C PROGRAM

Following is this month's example C function; it scans options from the command line.

Each call returns either the next command option letter, '?' to indicate an invalid option letter, or EOF to indicate the end of the argument list. The first argument is the argument count, the second argument is the argument pointer, and the third argument is a string representing the valid argument letters. The argument on the command line must be preceded by a hyphen for this routine. The original version of this routine is from Henry Spencer.

```
#include "stdio.h"

char    *optarg;        /* Global arg pointer. */
int     optind = 0;     /* Global argv index. */

int getopt(argc, argv, optstring)
    int argc;
    char *argv[];
    char *optstring;
    {
        register int c;
        register char *place;
        static char *scan = NULL;

        optarg = NULL;

        if (scan == NULL || *scan == '\0') {
            if (optind == 0) optind++;
            if (optind >= argc) return EOF;
            place = argv[optind];
            if (place[0] != '-' ||
                place[1] == '\0') return EOF;
            optind++;
            if (place[1] == '-' &&
                place[2] == '\0') return EOF;
            scan = place+1;
        }

        c = *scan++;
        place = index(optstring, c);
        if (place == NULL || c == ':')
            return '?';
        if (*++place == ':') {
            if (*scan != '\0') {
                optarg = scan;
                scan = NULL;
            } else {
                optarg = argv[optind++];
            }
        }
        return c;
    }
```

A call (from "main") might be the following:

```
    switch (getopt(argc, argv, "abcd"))
    {
    case 'a':
    :
    case 'b':
    :
    case 'c':
    :
    case 'd':
    :
    default:
    :
    }
```

Editor's Note: We goofed!

As usually is the case, we cannot stand prosperity. Normally we are lucky to get our regular column authors to get more than one or two months ahead. Most do get 1 or 2 ahead and I appreciate their effort. Rarely do I ever have to call to find out 'where is the column?'. But, 5 columns ahead?

Well, Bud Pass who helms this C' column was kind enough to get 5 (yep, 5) months ahead. We blew it!!! So you will find that this months column should have followed the column that was in the October '84 issue, which is the November column. But, the November column was the one that should have been March '85, and the between ones (December, January and February) should have followed the October '84, which was really the March '85. I think.....?

Anyway, we messed-up, I apologize. I appreciate your effort Bud, it is all our fault. Please keep us that far ahead and I promise to 'try' to keep 'em straight. And to all you who were scratching the ole noggin, well, again please accept our apology.

DMW

- - -

68000 USER NOTES

Phillip Lucido
2320 Saratoga Drive
Sharpville, PA 16150

UNIX

Anybody who has paid any attention to the microcomputer marketplace lately would have to work hard not to have heard anything about the latest miracle cure for the computer blues, the Unix operating system from AT&T. Depending on whom you listen to, Unix is either the best thing to ever happen for computer enthusiasts, or it is the unfriendliest, most user-hostile operating system around today. Just where does the truth lie between these two extremes?

Since Unix appears likely to be a big player among the 68000 operating systems, and because I have been using it extensively at work for the past three months, I've decided to spend this month giving you my own observations on the whole matter. Hopefully it will allow you to see through some of the hype surrounding the operating system.

The first thing to understand, I suppose, is that not all versions of Unix are identical. The Unix I use at work is a Microsoft 'port', or adaptation, called Xenix, which is running on an Intel 310 computer, which uses the 80286 chip, though there are Xenix versions or the 68000. Xenix is based on Unix Version 7 and System III, with some Berkeley utilities added. At least I think that's correct. You see, there have been several major incarnations of Unix since it began to travel beyond Bell Labs. The major ones directly attributable to AT&T are known as Version 6, Version 7, System III, and the latest, System V. To add to the confusion, Unix has been extensively worked over by people at the University of California at Berkeley, resulting in a version known as UCB or BSD V4.2, among others. On top of that, many Unix releases will include utilities written at Berkeley, but do not include the modifications to the kernel that are in a true UCB version of Unix.

What is there about Unix to attract such attention? There are three sides to an operating system that I will talk about in answering that. First, there is the internal structure of the OS, which is governed by the kernel and I/O drivers. Second, there is the user interface, by which a person can actually command the machine to do something. Finally, there are the utilities, those pre-written programs which come with the OS.

1 - The Kernel...

Unix was originally written for minicomputers, and is a powerful multi-user multi-tasking operating system. It seems to be oriented more at multi-person projects on a single machine than it is at controlling a personal computer, though that may change. In any case, one of the first things you notice about Unix is its size. It is BIG. The machine at work started with a 15MB hard disk, as well as 512K of RAM. The disk quickly proved to be insufficient, and we now are using a 40MB drive in its place. There are signs that the RAM may also prove inadequate, and we are thinking of adding another 512K. When the system b ts up, 140K of RAM is immediately used by the kernel, leaving 372K to share among all the users.

As long as you have a capable enough computer, Unix can do quite well at handling a number of people. The office machine routinely has seven programmers logged in, and response times degrade badly only when many of them start compiling at the same time. Since most of their time is spent in the editor, this is fairly rare. Be warned, though. We may be overloading our particular system. When it first started handling everybody at once, response delays of up to 10 seconds were not uncommon. This sort of delay can be unbearable, especially when using an interactive screen editor. That problem was alleviated by modifying the compiler, which compiles an in-house version of Basic we use, so that it ran at a much lower priority. I'm not sure if the delays were an inherent problem of Unix, or whether they were attributable to the device drivers written by Microsoft and Intel.

Unix does make some effort to cut down on one of the major reasons for delays, time spent waiting for a turn at the disk drive. First off, Unix really runs much nicer when using a large disk, or preferably, two disks, with system programs on one and data files on the other. The disks should have very fast access times. Given that, Unix does a couple of things to help the system along. For one, Unix sectors on disk are large, usually 512 bytes, 1K, or 4K long. Since most of the delay time using a disk is caused by track seeking, rather than actual reading or writing, performing disk operations in larger chunks can only help to speed things along. Also, Unix systems generally have fairly intelligent disk handling routines. The system keeps a number of internal buffers holding the latest sectors read or written, so that common requests, such as reading a directory, do not actually need to go to the disk. Also, when a program requests a disk write, the write is not completed before returning from the kernel to the program. Instead, the write is placed in a queue, which is arranged so writes and reads can be intermixed more efficiently. At most, the write will be delayed up to 30 seconds. There is an obvious problem here, in that a loss of power can result in the loss of data or damage to the logical disk structure, and in fact, disk crashes are annoyingly common with Unix, even if they are fairly easily recoverable.

A definite problem with Unix, at least as far as I have seen so far, is the lack of information needed to customize the system with new I/O peripherals. There is very little solid info on the writing of I/O drivers and the inclusion of new drivers in the kernel. The Unix documentation is not very helpful here. Related data are spread among six or seven different manuals, and are hopelessly sketchy at times. I have seen an Intel application note for writing device drivers for Xenix, but even that was only a four page snippet that left me wishing for more. If you intend to u e a Unix computer for software development or canned applications, fine, but if you need to hook some new lab machine into the computer, you probably better find the local Unix guru. This situation may not continue, and may not even be true for all Unix ports, since it may be up to the particular producer of the port, like Microsoft, Intel, Motorola, or AT&T, just what to release.

2 - The User Interface

Unix has sometimes been referred to as beginner-hostile, expert-friendly, and probably deservedly. It is a powerful system with many ways in which to combine commands. As such, it can be very difficult for a beginner to understand, while the power is very welcome to an expert user who wishes to perform some task with a minimum of interference from the computer. If you know how to use Unix, you can make it sing! Get a little careless or a little lost, though, and you'll quickly be searching for those backups you made.

Much of the Unix user interface resides in the shell, which performs the same function as the shell in OS-9 or the command line in Flex, that of passing commands to the kernel to begin execution of programs. The Unix shell is capable of much more, though. Instead of simply dealing with commands line by line, it is a full programming language, with for, while, and repeat loops, and a case statement, much like the same constructs found in the C language. There are named variables, parameter substitution, and conditional execution of commands. For instance, consider the following short example:

```
for name in joe bob jim george
do
   cp file /usr/$name
done
```

This short program instructs the shell to assign to the variable 'name' the values 'joe', 'bob', 'jim', and 'george', one at a time, and perform the instructions inside the do..done loop for each assignment. The cp instruction says to copy a file na d 'file' to a directory whose title is formed by concatenating '/usr/' with the current contents of the variable 'name'.

This example is really very trivial, but the important point is that to do something like this, with looping and parameters, I do not need to type text into a command file, which I then execute. Instead, I just type each line, one at a time. The shell recognizes the fact that the for l p is going to require multiple lines, and waits till I type the terminating 'done' before executing the entire program. Also, with the power available in the shell, many programs on the disk are actually just shell script files, instead of being compiled object C object code.

The shell also allows you to use I/O redirection and piping, with somewhat more variants than are available with OS-9. The ability to hook the output of one program to the input of another, which is piping, is something I would not want to live without, now that I have used it. It is simply astounding how much can be done by chaining a number of small filter programs together.

There is another shell program commonly available with Unix systems, known as the c-shell. This is one of those Berkeley utilities. It has the same looping and conditional constructs as the standard shell (which is also called the Bourne shell, after its author), but in a syntax closer to the C language. The c-shell can also remember recent commands which are typed in, and can re-execute or edit them to create new commands lines or correct typos. The lack of this is a problem with the standard shell, which doesn't have *any* ability to re-use command lines, as you can with OS-9 using the ctrl-A key.

Obviously, I like the Unix *shells*, since I have managed to get past the beginner-hostile phase, even if I am not quite an expert yet. I have heard many people complain about the bewildering complexity of OS-9. Well, take that and square (cube?) it to get some idea of the problem with Unix and it's shells. You can certainly get along issuing a single command per line, but the true utility of the system isn't revealed until you learn, learn, learn.

3 - The Utilities

Now I come to what is probably the heart and soul of Unix. There are plenty of operating systems which can efficiently drive a large system with many users. There are also systems with powerful command line abilities. But none that I have seen have the wealth of programs available in Unix.

Unix comes with an assortment of about 200 different utilities. Many of these are quite small, performing a single function such as word or line counting (wc), sorting (sort), or searching through files for instances of text patterns (grep). These small one-function programs, when chained together with pipes, can perform marvels.

Unix doesn't stop there, though. There are medium size

programs which can be used to perform many different functions. Some of the handiest are programs like sed, awk, and make. Sed will look through a text file, line by line, and perform an editor-type operation on each line matching some condition. The lines are then sent to the standard output, where they may be piped to other programs. Sed is very handy at picking only certain lines from a file, and then selecting only the *required* fields from those lines. Awk is another pattern matching, line oriented program. It is much more capable than sed, though, with the ability to accumulate data from selected lines and output it after the entire file is traversed. Awk is in fact an entire programming language, patterned after C, with arrays and looping. Even better, the arrays are not necessarily indexed by numbers, but can be indexed by words (this is known as associative addressing). With awk, it is a simple matter to count the number of occurrences of each word in a file. The last program I mentioned, make, is a lifesaver when you are working on a large software project. If you have ever written a program which was split into a number of different modules, you know how easy it is to forget just what portions need to be recompiled after a change is made. With make, you create a list of dependencies, on the order of "file1 depends on file2 and file3, while file2 depends on file4." If you modify file4, make will recognize the fact that file1 and file2 must be recompiled, while file3 need not be.

Finally, there are the large utilities. Unix comes standard with a C compiler (of course), but in addition there are programs like lint, lex, and yacc. Lint is a program which will go through your C program, and detect any questionable lines, such as calling a function with the wrong number or wrong type of parameters, or assigning a pointer value to a pointer variable of the wrong type. C is notorious for letting these sorts of things slip through, since after all much of the power (and danger) of C is the ability to play tricks with your code. Lint makes sure that any tricks in your program aren't tricks on you, too. The programs lex and yacc are two examples of programs which write other programs. Suppose you want to write a calculator program, which accepts input lines like '2 + 3*(4 + 5)', and gives you the answer back. Such a program is basically a compiler for mathematical expressions. Compilers can be quite lengthy and difficult to write, though. With yacc, all you have to do is specify the format of the language to be accepted by your compiler program, along with what actions are to be performed based on the lines in that language. Yacc will then write your compiler for you, in C. Lex does a similar job, creating programs known as lexical analyzers, which perform a function found in compilers. Both these programs can make writing a big program much, much easier, so it becomes possible to design small custom languages for dedicated applications.

An interesting thing is now occurring among the other operating systems. As more people are exposed to Unix and its *utilities*, they are becoming unsatisfied with the programs that come with their own computers. In response, more and more packages of the various utilities are being written, independent of Unix, and sold or placed into public domain. For instance, Microware is issuing a number of programs which are derived from Unix utilities, and a number of the utilities have been written by OS-9 programmers and placed in the Users' Group Library. I may even submit a few goodies there, if I can only track down a few pesky bugs! I wish, though, that some of the larger programs were so available. I would very much like to have versions of make, lint, and yacc.

Putting It All Together

In the end, whether Unix is a good idea depends on your circumstances. If you are heavily into programming projects, then Unix can make your job a lot easier, especially if you take the time to study the manuals and learn the utilities. For canned applications or computer-naive users, Unix can be a problem, since it is so complex. Also, since Unix has spent most of its life as a minicomputer OS, few microcomputer applications are available, though more are being written constantly, now that Unix seems to be getting the big push. Also, Unix is not really set up for real-time applications, though some work is being done there. The thing to remember, I suppose, is that Unix was written by a bunch of computer scientists, for computer scientists.

Finally, Unix really is very, very big. Some systems are being sold with Unix for prices of only $5000 or so, but these don't seem to have all of the utilities or a large enough hard disk. To run Unix, you probably need a system costing at least $10,000, with at least 20MB of hard disk space and 512K of RAM. Unix can run with less, but only grudgingly. There are cheaper alternatives, with various operating systems heading for different niches from those held by Unix. And, with any luck, somebody out there will write a good version of yacc that I can use.

Enough. Hopefully I've *managed* to be unbiased and even here. If I've gotten something dreadfully wrong, let me know. After all, three months with an operating system hardly makes you a Unix guru.

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

The OS-9 Seminar and More RPC Thoughts

I was a little disappointed in the OS-9 Seminar this year. Most of the hobbiest feel has gone out of the thing. Microware is now a big company with a beautiful new building. The seminar was a slick, professional production. I'm not a slick, professional person; I liked it better with the rough edges.

The smooth production notwithstanding, there were some important high points. I watched OS-9 68K in operation. It is impressive! Establishing a good speed comparison between OS-9 for the 6809 and OS-9 68K will be a major project, but I saw a qualitative difference. Especially when crunching numbers, OS-9 68K ran LOTS faster. Of course, the machine running OS-9 68K was big. I don't remember the manufacturer, but the computer would have made a respectable student desk. It used the SUN processor architecture and I think it had a megabyte of memory.

The large address space available with the 68000 lets Microware add power to the utility programs. The utilities with OS-9 68K are quite UNIX-like. The enhanced shell is particularly nice.

Microware did a good job of making OS-9 68K software look good at the seminar. The hardware vendors helped by showing 68000 based machines. Smoke and Hazelwood were ready to sell 6000(0,8) SS50 machines. For those lucky enough to have Helix computers, there is an upgrade that will let you run either a 6809 or a 68000. You can't select a processor in software, but a switch on the front is better than nothing.

I'm deeply disappointed in Gimix. I figured they would come out with one of the world's best 68000 machines. My experience with my Gimix gave me confidence that their product would be exceptional. After some serious consideration Gimix has tentatively decided not to produce a 68000 board for the SS50. Their business reasons seem sound to me, but I can tell you I'm disappointed. I haven't entirely given up. Richard Don (from Gimix) hasn't entirely ruled out a 68000 for my computer, just dashed my hopes for the near future.

There is some other tentative good 68000 news. Fujitsu sells OS-9 machines in Japan. Their 6809 machine is and will probably remain only available in Japan, but there is a good chance that their

68000 machine (the 16s) will be sold in this country. I'm excited about the possibility.

My beloved computer doesn't impress my PC-owning friends. I don't have a bit-mapped color screen (except on my CoCo). My processor has a miserable 64K address space. I can't even get a mouse (except for my CoCo). For this machine I payed more than any of my friends payed for their computers. Put simply, my Gimix isn't a consumer machine.

The Fujitsu is a consumer machine. It has excellent color graphics, windows, a 16 megabyte address space, a mouse, and a price that looks good compared to a PC. You can get one with an 8086 now. There are lots of rumors suggesting that the 16s with a 68000 board and OS-9 will be available soon.

The Fujitsu is an early sign of the OS-9 activity in Japan. I wouldn't be surprised if there were more OS-9 users in Japan than in this country. They have a Users Group with a slick, typeset newsletter that makes ours look silly (if you're thinking of joining, its in Japanese.). I hear from Microware that OS-9 is now the number two operating system in Japan behind UNIX.

Another popular subject for rumors was the CoCo. I've heard lots about a new super-CoCo. I expect most of what I heard is wishful thinking, but here's a summary.

Wouldn't a CoCo with a DAT (Dynamic Address Translator) and enough memory to run OS-9 level Two be nice. A machine that powerful would certainly deserve a real ACIA (serial I/O chip). That new graphics chip set from Motorola would be a nice addition. If they priced it close to $500 it would probably sell like hotcakes.

The more extravagant rumors have Tandy selling a 68000 machine with OS-9 68K.

The closest I have to official confirmation from Tandy about any of this is that when I asked a salesman at a Radio Shack Computer Store about the Level Two CoCo he told me that he would be strung up if he talked about it -- wait for the official announcement ... but what could I tell him about OS-9 Level Two?

There was talk at the seminar about the fine, low prices Radio Shack has for Microware software. People without CoCos wonder why they should pay about double what Radio Shack charges for each piece of software. There are lots of reasons with Tandy's massive buying and selling power probably being the most important. The difference that effects us (other than in the pocket) is that software that comes from Tandy comes with only Tandy support. If you buy software from Microware or some licensee other than Tandy, you get 90-day's support from the Microware hotline and the option to buy continued support. I can tell you from experience that the Microware hotline is a useful service. Microware is still small enough that you get to talk to a real programmer, perhaps even the person who wrote the software in question.

Additional thoughts about Last Month's Column

If you've had time to think about the remote procedure call software I presented last month, you've probably decided that it is an interesting piece of special purpose code, but not all that useful.

It was designed for two purposes. It is a way to run large programs -- programs with, say, 150K of code -- as a high-speed alternative to overlays or chaining. It is also the basis for a powerful inter-machine communication method. It was originally written for communication, but I never finished the rest of the system. It's still sitting on my list of unfinished projects.

The main problem with RPC is that it is a neat package. There are no options either to be confusing or to add power. Here are some suggestions for interesting options you (or I) might add:

Make the third parameter in the call to RPC be the name of a shared data module. Sending data to and fro through pipes uses substantial processor time. A data module can be shared between two processes almost free. There could be trouble if several processes were allowed to write into the shared data module at the same time, but RPC doesn't return control to the calling program until the remote procedure completes.

Separating the part of the RPC subroutine that waits for the remote procedure to return from the part that fires it up would permit the calling procedure to continue running together with the remote procedure. A nice use of a version of RPC that permitted concurrent execution would be to do housekeeping while I/O was going on. Something like doing garbage collection while waiting for input.

Dividing the rpc subroutine into three parts: startup, send, and receive, would complete the job. Firing up a procedure takes lots of time. Getting it going once, then just sending it more data every time you want to use it would save time.

There is a ceiling on the number of remote procedures that can be handled this way. There is a limitation on the number of paths available to a process. After the standard I/O paths their are only 13 left. That's enough for two pipes to each of 6 remote procedures with one left over. Realistically you'll need at least a path or two for RPC overhead and one or two for disk I/O; five concurrent remote procedures is a reasonable ceiling. Once you have a remote process started up you have to keep the paths containing pipes to it available somewhere.

There is a nice formalism that can be used with a version of RPC with startup separated from parameter transfer, the coroutine. The essence of the thing is that the return works like a call. After a procedure returns control to its caller it waits frozen in place until it is called again, then it continues from the point it returned from. This sounds tricky, but with control transfer through pipes, it's hard to avoid. Call the remote procedure by writing the arguments to it. The remote procedure waits for the arguments to arrive in standard input, processes them, writes the arguments out standard out, and does another read for more arguments. The caller reads the returned arguments, and does some processing, then it might call the remote procedure again by writing more arguments to it (in which the remote procedure would pick up from the read at which it was waiting.)

Let me give a little plug to the compilation of these columns that I put together this summer. Don Williams is publishing it for me. There's nothing important in it that hasn't been in this column. It's my first year-and-a-half of columns collected together, but I'm rather proud of the format. The print is larger than is used in '68' Micro Journal. The programs do particularly well with larger print. I added some footnotes where they seemed important, and used tricks like boxes, tables, and figures where they belonged. The most important

feature of the collection is the index. I skip from subject to subject each month. The index makes the collection much more usable.

* Editor's Note: Just as we were putting this issue to bed. I received a call from Richard Don of GIMIX. He was calling to let me know that GIMIX will do a 68000 board for their GIMIX III S50 Bus System, after all. Actually it will be a 68010.

What is a change though is that it is being designed especially for the new TSC UniFLEX VM. As to the status of a GIMIX 6800(?) CPU for the OS-9 operating system, I am guessing, but I would bet that there will be one also for the OS-9 68K when level 2 rolls off the production line at Microware.

The TSC UniFLEX VM is already out in the field, it is running on the Tektronixs micro system. But in this case it appears that a GIMIX CPU card is being developed for one particular operating system, where as before, almost always, the operating system was designed to the card. If it will also be an effective OS-9 card, only time will tell.

My 'gut' feeling is that GIMIX will do a 6800(X) something or another for OS-9 68K level II on their GIMIX S50 Bus level III machine, if not this project. Or in other words, I will bet that there will be both disk systems running on the GIMIX S50. I don't believe Richard Don and GIMIX will let a sure bet slide by. But, if they don't, I imagine some other enterprising manufacturer will, but my bet is on GIMIX to support their own product to the fullest.

UniFLEX VM is supported by 'demand-page virtual memory'. Virtual memory, simply stated, allows programs larger then normal RAM to run as if there were more RAM than actally exist on the system. The difference between the version of UniFLEX you probably know is this; non-virtual memory operates by 'swapping task' from RAM to disk, however, the 'task' must fit in existing RAM allocated for that task or installed in the system. Virtual memory can run one or more tasks that are larger then existing RAM in the system. Normally, non-virtual systems swap the entire task when running, virtual memory systems swap only a portion of a programs code or memory.

The GIMIX UniFLEX VM 68010, can support 1 megabyte of RAM. Demand-page UniFLEX VM enables this hardware to run programs that require up to 4 megabytes of memory. Something that normally is left to much larger (and much more expensive) minicomputers and large mainframes.

DMW

- - -

SINGLE BOARD COMPUTERS-6809

Part Three - The Compacta 6809 "UNIBOARD"™

Final

A short recap

Recently we received for review 3 different single board 6809 computers. All three are 64K systems, with 56K standard per FLEX™ convention. All three boards run FLEX, two have also licensed OS-9™ level one. The two FLEX systems recommend that you purchase FLEX from your favorite source and use their modified drivers. Essentially this requires most any FLEX.COR and append the drivers to make a bootable FLEX system. Some consideration should be given to certain SWTPC FLEX versions, however, all can be made to work. S ecifics will be covered in the review of each system.

The three systems we will look at are:

1. The PT-69™
   Peripheral Technology
   3760 Lower Roswell Rd.
   Marietta, GA 30067
   404/973-0042

2. ST-2900 System™
   Sardis Technologies
   2261 E. 11th Ave.
   Vancouver, B.C.,Canada V5N 1Z7

3. The Compacta 6809 "Uniboard"™
   Digital Research Computers (of Texas)
   P.O. Box 461565
   Garland, TX 75046
   214/271-3538

Notice should be taken that we will review each system in the order of A-Z. Why? Well they all have certain strengths and weaknesses, as we see it. Also we ended up having no particular favorite, as each has certain merits not available to the other two. All three are advertised in 68 Micro Journal and are running either in our offices or our lab (meaning they have been tested and accepted by our standards). All three perform well. Any one of the three when combined with disks and a CRT or keyboard and monitor (depending on the system) make an excellent, general purpose or specialized 6809 64K computer. The boards alone make great and very economical 6809 controllers or stand-alone systems. I see an upsurge in 6809 activity due to the economy and availability of these systems running all the popular 6809 disk systems and software!

I have long seen the need and attempted to get some of our present 6809 computer manufacturers to make a similar system. A very accurate survey some two years ago indicated that many of you wanted such a system. Only SWTPC and WaveMate have done so.

WaveMate blew it by making the hardware and software dependent on a double density disk directory for FLEX. All normal FLEX systems use single density directories, for both single or double density format. Had they listened I sincerely believe that they would have had a winner, but now only SWTPC advertises a system in a desktop configuration (X-12+), and I understand it is doing quite well. However, by utilizing a CRT terminal similar to the Heath H-19, which has provisions for disks also, the entire system can be in one package. And that is the wave of the future, something we should have done years ago. Now with two of these, desktop complete systems are possible. With the other the size of the board is slightly too large, due to features not available on the other two. Remember, I said advantages and disadvantages.

Now, as to the Heath H-19, it is no longer in production, but many are advertised as used and at very good prices, so it should not be too difficult finding a low price used one or a similar type. Should any of you out there have a used Heath H-19 for sale, please let me know as I am certain I will be receiving many inquiries for availability of used ones.

The Compacta "Uniboard"

The other two boards in this series are rather small and fit handily into the Heath H-19 CRT terminal. This board would not. It didn't miss by far not quite. That is its biggest disadvantage. It has more utility and power than the others. It has DMA (Direct memory access) for the disk system and bus structure. DMA allows far faster data transfers. Also this system allows for 4 5" disk drives, 4 8" disk drives and the documentation claims a hard-disk is available from Compacta. THe DMA, 8" and hard-disk facilities put this board more in the class of the 'big boys'.

The board comes as a kit only. It can be purchased as a bare board, but including the PAL (Programmable Array Logic) devices (2) and the monitor, or it can be purchased as a complete kit with all parts (except -12v) switching components, which are sold for a slight higher price if your power supply does not supply -12v at about 100 ma. The PAL devices which can be inhouse programmed for a variety of functions give the seller the ability to literally design his own personalized LSI. Custom programmed PALs are available from Digital Research Computers: (of Texas) or Compacta.

The board size is 11 X 11 1/2 inches and is high quality glass epoxy, double sided, plated thru. The device count on this board is approximately twice what the other two contain. The only improvements that could be designed into this board that would make it smaller is that it presently uses 16K memory devices. Remember, this board has been available longer than the others and I am told that it will be upgraded in the future to take advantage of the more dense memory chips coming to the market. It also will probably have provisions for extended memory addressing.

This system runs both FLEX and OS-9 level one. Future versions should allow OS-9 level two capability.

Documentation

The documentation, like the others is not Heath quality. It is adequate and also includes a full schematic diagram as well as parts list and even timing charts. We found that the setup and alignment procedure section sufficient to allow a fairly inexperienced builder to get it running without unnecessary hassle.

The only problem experienced was that there seems to be a 'bug' in the seek routines in the monitor that fails double sided operation of the disks. A fix on this as well as other improvements are scheduled to be published in 68 Micro Journal in the near future. Also the distributor promises a fix soon (probably by the time you read this review). Aside from this we found the system runs very smooth.

The documentation consists of the construction and alignment manual, the monitor user's manual and the operations manual. THe monitor is one of the most extensive disk functioning monitors we have seen, in addition to most normal monitor functions (see later).

All in all the average builder will find little fault with the manual, even the added notations that bring it current.

The "Uniboard" Monitor

The monitor contains all of the routines to support the FLEX disk operating system. The following is a list of the basic monitor command repertoire:

G - Go to address <start> <end>
A - Alter memory <address>
F - Fill memory <start> <end> <constant>
M - Move block memory <start> <end> <destination>
D - Display memory <start> <end>
T - Test memory <start> <end>
R - Read disk sector <drv> <trk/sect> <address>
W - Write disk sector <drv> <trk/sect> <address> (R/W in 256 byte blocks)
B - Boot normal FLEX
U - Boot unknown FLEX
L - Boot general FLEX

For the R/W disk sectors functions the monitor returns the following errors

READ/WRITE	Seek/Restore/Select
Drive Not Ready	Drive Not Ready
Write Protected	Seek Error
Write Fault	CRC Error
ecord Not Found	Can Not Restore
CRC Error	
Lost Data:	

We will not go into the normal monitor routines here as they are standard in their functions. However, a short explanation of the boot functions is in order, and are as follows:

BOOT NORMAL FLEX: The boot loader is read in from track 00, sector 01, and control is transfered to C100 (the loader) This load routine is one supplied for this board.

BOOT UNKNOWN FLEX: This command loads non-unIboard versions of the loader and I/O routines. The disk must be linked to FLEX. That is, the starting tr ck and sector must be in bytes 5 and 6 of track 00 sector 01. The onboard loader is then called using the track and sector at location C105. Control to FLEX is then transfered to location CDDD.

BOOT GENERAL FLEX: This command expects to find the FLEX system at track 01 sector 01. Drive selection is made and the loader is called in. Once FLEX is loaded then the I/O and disk drivers and other necessary information is passed and made known to FLEX at CDDD. The general version of FLEX from TSC can be loaded by this command. Also STAR-DOS+ from STAR-KITS and Data-Comp can be run, exactly as FLEX does.

Additional monitor routines are available to the user thru a vector table. They interface the system console, the disk system, printer interface and the interval timer used for spooling.

The monitor uses RAM from EE00 to EEFF. The system stack is set to EDFF. Locations EEDD to EE48 are used by the monitor for housekeeping and general I/O functions.

General Features

The system in addition to the features outlined ab e has the following additional features:

6809E processor
64K RAM
4K ROM/EP OM
Video controller with programmable formats
Parallel keyboard interface
Parallel (Centronics type) printer interface
Dual timer for real time clock and measurements
Serial interface, both RS232 and 20 ma
Expansion interface with DMA channel

The circuitry can be divided into ten functional blocks.

Timing is from a master 32.0000 Mhz xtal oscillator. This is divided into a 16 Mhz clock for the video dot timing and a 1 Mhz clock for the system E clock. The divider also produces the CPU Q clock.

CPU cycles are interleaved with video display cycles. The CPU accesses memory on the positive portion of the E clock and the video accesses on the negative portion. The 6845 video controller accesses RAM two bytes at a time to insure 80 or more characters per line (speed considerations). RAM is effectively 8 bits wide for the CPU and 16 bits wide for the video.

Two PALs are used. Address space decoding is done by PAL logic on the chip select strobes. A second PAL controls the column address strobes CAS for the RAM. In addition it also generates the enable signals for the RAM bidirectional buffers and the Read FDCRD and Write FDCWR strobes for the disk controller - 1793/8877.

The RAM is set up as four banks of 16KX1 dynamic memory. The ROM/EPROM can be either a 2716 or a 2732 device. A PAL handles chip select for this device.

Up to four 5" disk drives and four 8" disk drives can be controlled by the system simultaneously. Any mixture of size and density is fine. All data transfers are by the DMA controller 6844.

Parallel interfacing is done by a 6522 Versatile Interface Adapter VIA. One side interfaces the keyboard and the other is set up as a Centronics type printer port. The VIA also include two timer/counters with 16 bit resolution. The printer can be programmed for polled or interrupt driven operation.

A 6551 ACIA interface is available for printer, modem, etc. operations with the system. The on-chip buad rate generator generates 15 standard buad rates, xtal controlled. In addition 20 ma, active or passive, loop interfacing or RS232 interfacing is jumper selected.

The video 6845 generator controls the interfacing to a raster scan device (video monitor). A character ROM is used by the 6845 for dot patterns for ASCII characters as well as some graphics symbols.

Two separate video interfaces are provided. One for composite video and one for separate TTL video and sync.

DMA is accomplished by a 6844 DMA chip. All disk data transfers are by DMA. The DMA chip is actually a microprocessor with very specialized functions. It takes control of the bus from the main 6809 CPU and then effects data transfers between memory and the disk system. High data transfer rates are made possible, much more efficient than normal I/O type transfer as used by the other systems.

The documentation indicates a winchester hard-disk is available for this system, however, since they did not include one with this unit, can't tell you much (why didn't you, Jim?)

An 'expansion bus' is included. This bus has most all the necessary signal and control lines to interface the board with practically any type of device. Peripherals may access RAM and other on-board resources thru this bus and may even use the DMA features of the system.

With the cost of the **Unlboard**, a CRT monitor, keyboard, power supply and two 5" disk drives DD DS, the total system cost should be $1500 or less, and that is the advantage of the new wave of single board 6809 computers, cost, compactness and semi-portability.

For those applications demanding additional I/O, and other specialized peripheral interfacing then one of the larger S50 Bus system may be required, but for many this is the way to go. And it is good for the industry, for experience has shown that satisfied small system owners eventually graduate to larger and more complex 6809 systems rather than go off to the 'other side' (who wants to learn new languages, buy new software and essentially start all over?) And for many this can be either a 'big' or 'small' system.

Price $329.00
Complete kit - fully socketed

Bare Board w/PAL's and two EPROMS **$99.95**

See **DIGITAL RESEARCH COMPUTERS (of Texas)** Advertising for additional specs and ordering information.

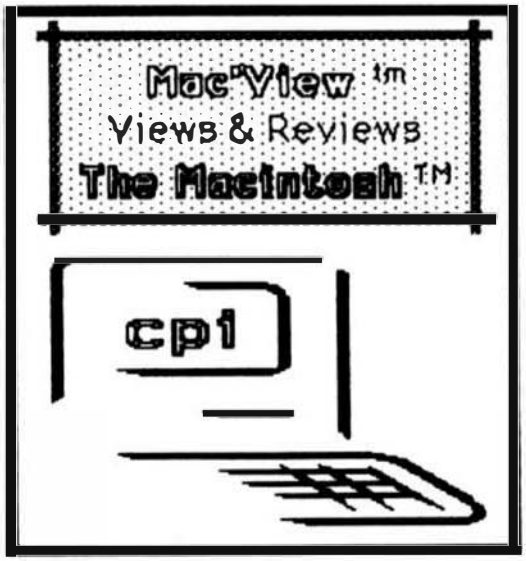
This completes our three series review of all the single board 6809 computers we have received todate. I have a strong feeling that there will be more to come. Also I am told that all will be offering either add-on memory expansion or new boards with extended memory addressing.

From the look of things to come the 6809 is far from dying. Despite all the neglect it has received from about everyone but us. I like it, the CMOS version when it becomes readily available should rekindle a lot of interest in the 6809. We have a long way to go with this device. In so ways it is superior to some more exotic devices (680XX, etc).

For those wandering off to fight the wars of competition on the 'other front', I expect some to return, others would probably desire to also, but the fierce struggle 'over there' will have been too much. They will succumb and we will all be loosers as a result. For us, we intend to stay. Sure, we will cover the other newer devices as they come and have something going for them, but for a long time to come, the 6809 will reign.

DMW

- - -



For over 6 years now we have brought you more information on the Motorola 68XX(x) CPU systems and applications than **all other computer magazines combined!** And starting this month we will, as material and space permits, bring you the same sort of treatment for the Macintosh.

We are receiving a lot of interest from our present readers, and some new Mac readers concerning our covering this system. Some already have a Mac, and some are having a Mac *attack*. Especially after they have a chance to play around with one. Everyone seems to go overboard about the graphics. Nearly every day I get a call or letter from some reader asking questions about the Macintosh. Seems the many feel that because we do so well with the other 68XX(X) systems that this one is a natural extension to our usual coverage. So I guess it will be.

As the months pass you will find that we will expand our coverage of this as well as several other more popular 68000 machines. But I must warn you now that without your input it won't amount to much. We have done well together, you sharing with the thousands of our other readers and they sharing with you, to the extent of their ability. We need input from both pros and beginners. Surprising at the amount of useful information we have published, over the years, from hobbyist and beginners. And I know, many, including myself, have learned a lot from the experiences and input from those not considered professional. I remember when, in our ranks, there were no pros. And not too long ago either.

Now a little about what we have here at CP1 and 68 Micro Journal in the way of Mac stuff. Not much. Why? Well because simply there just isn't too much available. We have the Mac, of course, and also the Mac Imagewriter and an external 3 1/2 inch drive. Several software packages (not including those that we got with our Mac), which we will be reviewing soon. But, as of now, there just isn't too much software or hardware, from independant sources available. As it becomes available we will pass reviews and what have you, on to you, via this column.

I trust we will soon have the memory expansion unit installed in ours as the 128K of RAM sure goes away in a hurry! Also the addition of a hard-disk to our Mac will certainly be done justas soon as we can get one. So we are all in the same boat, a fine system, but nothing much in the way of soft/hardware support yet. From what I am hearing it won't be long. Seems sorta like where we were 4 or 5 years ago. Good basic hardware, but little else. Now look at what is available, and I believe that because of the number of these things sold, the shortage won't be for long. Maybe Apple will get around to double sided drives (been around now a while), as 400K per drive, and a memory hog sure strings things out. However, all in all, it is a state-of-the-art hardware package, mostly.

So, the ball is now in your court. I will do what many of you are asking for; a Macintosh column, but I need your **input!** And **PLEASE** don't be bashful. Even if you feel that it might not amount to much or may seem trivial to you, remember, there are thousands of folks (our readers) who are just as hungry for information about the Mac, as you probably are. With all of us sharing, it could be another good thing, for all of us.

One thing for sure, after reading a couple of the other Mac supporting magazines, everyone is looking for information. And that, my good friends, is what we have always been about. Again, we won't be the fancy or slick type, but we sure will let the shoe drop straight.

So lets get started NOW!

CRUNCH COBOL

COBOL - An Example Program

I enjoyed reading Ron Andersons review of "CRUNCH COBOL" in the October 68' Journal, and sympathised with some of his problems with a new language. To round off his review I have prepared a program example for the upper/lower case conversion exercise which Ron tried in COBOL. As Ron realised, COBOL techniques are not the same as other languages.

The first thing to do when writing a COBOL program is to switch off the computer! In other words, design the program first then code and test it.

During my commercial programming days I was trained in "modular programming". This technique is widely used, and is ideally suited to a language like COBOL. As the name implies, modular programming implies that the problem is broken down into independent procedures. These procedures are controlled by higher procedures, until the highest level is reached, which is the control module. Another name for this is "top down" design.

The general idea for this program is that it will read a record, inspect the contents replacing upper case by lower case, write an updated record and repeat the process until an end condition is met. The block diagram for this program looks something like the following :

A-MAIN
Control Section

B-OPEN	C-PROCESS	D-CLOSE
Initialisation	Main Update Sect.	Closedown Sect.

CA-INSPECT
Character shuffle

My own rules for block diagrams are :

1) A higher level may only access the next lower level. For example A-MAIN may not directly call CA-INSPECT.

2) Dependent modules *begin* with the same letter as the parent module. So CA-INSPECT belongs to the C-PROCESS or *update* section of the program. The exceptions are the highest level which I always call A-MAIN, and the lowest level which are common library routines prefixed by "Z". You can fool around with the naming convention until it suits you - then stick to it!

Looking at the diagram it could represent any general update program, in fact programs like people are largely variations on a theme (vive la difference!).

Alright, that is how we will code the procedure. But before we can do that we first have to consider the data requirements of the program.

There are going to be two files accessed by this program, the input file and the output file. As Ron realised, COBOL is record orientated so we have to decide the maximum sizes of the input and output records. A reasonable size is 80 characters. The code to do this is shown below :

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT IN-FILE ASSIGN TO FILE-1
        ACCESS MODE IS SEQUENTIAL.
    SELECT OUT-FILE ASSIGN TO FILE-2
        ACCESS MODE IS SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
FD  IN-FILE.
01  IN-REC       PIC X(80).
FD  OUT-FILE.
01  OUT-REC      PIC X(80).
```

IN-FILE is the name of, I hope, the input file and the FD (File Descriptor) clause tells me that it has one record, IN-REC of size 80 ch racters. The same is true of OUT-FILE. The actual usage of these files in terms of input and output is determined by the program code.

Now we have our files, what about any temporary storage? If this were a long program then one would fill in temporary storage as required during coding, but in this case life is fairly simple. Firstly we need tables for comparisons of characters, "A" thru "Z" and "a" thru "z". In addition we need one counter. The working storage is shown below :

```
WORKING-STORAGE SECTION.

 01   LOW-LETTER PIC X(26) VALUE
"abcdefghijklmnopqrstuvwxyz".
 01  LOW-REDEF REDEFINES LOW-LETTER.
     03  LOWER  PIC X OCCURS 26.

 01   UPPER-LETTER PIC X(26) VALUE
"ABCDEFGHIJKLMNOPQRSTUVWXYZ".
 01  UPPER-REDEF REDEFINES UPPER-LETTER.
     03 UPPER    PIC X OCCURS 26.

 01  I          PIC  99 COMP.
```

What we have above is an array LOWER(26) containing "a" thru "z", an array UPPER(26) containing "A" thru "Z" and a counter I.

If by now you get the impression that the "actual" program is not so significant in COBOL you are right! Careful design and planning reduces the complexity of COBOL procedure code from the sublime to the mundane.

Let's begin with the main section :

```
A-MAIN.
     PERFORM B-OPEN.
     PERFORM C-PROCESS.
     PERFORM D-CLOSE.
A-999.
     STOP RUN.
```

Fairly *painless*, and largely reusable. The main program calls three sections in turn and then returns control to FLEX. Now let us progress alphabetically through the sections :

```
B-OPEN SECTION.
B-000.
     OPEN INPUT IN-FILE.
     OPEN OUTPUT OUT-FILE.
B-999.
     EXIT.
```

As you might gather, here we open IN-FILE for input, and OUT-FILE for output.

B-OPEN is a SECTION. When B-OPEN is performed by A-MAIN, then control is returned to the calling routine by the EXIT statement. Within B-OPEN there are two PARAGRAPH's namely B-000 and B-999. These paragraphs can be accessed by name. for example the line :

```
     PERFORM B-000.
```

would execute both the OPEN statements in the above code. If several paragraphs are involved then the syntax is :

```
     PERFORM X-010 THRU X-100.
```

which will perform all intervening statements from X-010 until the end of paragraph X-100. Beware - if your program code doesn't get to X-100 then strange things will happen!

When a section is compiled independently in CRUNCH COBOL then only the SECTION names are available externally at link time, so paragraph labels can be "local" to this extent.

```
C-PROCESS SECTION.
C-000.
     MOVE SPACES TO IN-REC.
     READ IN-FILE AT END GO TO C-999.
     PERFORM CA-INSPECT VARYING I FROM 1 BY 1 UNTIL I =
26.
     WRITE OUT-REC FROM IN-REC.
     GO TO C-000.
C-999.
     EXIT.
```

This section performs the dependent section CA-INSPECT 26 times, incrementing the counter I e ch time.

It is possible to write COBOL without "GO TO" statements but I'm not picky. This section gets a record from IN-FILE, processes it and writes the output record. Note that COBOL reads FILES and writes RECORDS. This is because files may have many record types which overlay the same buffer area. The read verb fills the input buffer with the next record, it is up to the program to sort it out. Usually a record identifier is put in the same place for each record type.

```
CA-INSPECT SECTION.
CA-000.
     INSPECT IN-REC REPLACING ALL UPPER(I) BY LOWER(I).
CA-999.
     EXIT.
```

CA-INSPECT is the "meat" of the program. The COBOL inspect verb is used to replace all occurrences of the current character UPPER(I) by LOWER(I).

```
D-CLOSE SECTION.
D-000.
     CLOSE IN-FILE OUT-FILE.
D-999.
     EXIT.
```

The closedown section just closes both files.

Thats it. Now we can switch the computer on and type the program in with a fair chance of success.

You could write this particular program using many fewer procedure lines, the intention here is to show you how to structure programs for maximum effect. No one section is too difficult to comprehend, and each section can be tested easily. This program runs somewhat faster than the time quoted for Rons attempt at this program. There IS a much faster way to program this conversion. It involves a typical "sneaky" trick in COBOL using multiple redefinitions of data areas. Consider the following :

```
01 W-DATA     PIC 11 VALUE ZERO.
01 W-DATA-REDEF REDEFINES W-DATA.
  03  FILLER PIC X.
  03  W-CHAR PIC X.
```

W-DATA is a 16 bit unsigned binary field initially set to zero. It is redefined as two characters. So the update paragraph could look like :

```
C-PROCESS SECTION.
C-000.
     MOVE SPACES TO IN-REC.
     READ IN-FILE AT END GO TO C-999.
     PERFORM C-100 VARYING I FROM 1 BY 1 UNTIL I =
80.
     WRITE OUT-REC FROM IN-REC.
     GO TO C-000.
C-100.
     MOVE IN-CHAR(I) TO W-CHAR.
     IF W-CHAR NOT < "A" AND W-CHAR NOT > "Z"
       ADD $20 TO W-DATA
       MOVE W-CHAR TO IN-CHAR(I).
C-999.
     EXIT.
```

Having been through all this you might assume that the complementary program just requires a transfer of tables. Right, and also wrong. It is a common requirement to go from lower case to upper case especially for file keys. So there is an INSPECT syntax in CRUNCH COBOL to handle this and the full program is listed below :

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT IN-FILE ASSIGN TO FILE-1
         ACCESS MODE IS SEQUENTIAL.
    SELECT OUT-FILE ASSIGN TO FILE-2
         ACCESS MODE IS SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
FD  IN-FILE.
01  IN-REC       PIC X(80).
FD  OUT-FILE.
01  OUT-REC      PIC X(80).

PROCEDURE DIVISION.
A-MAIN SECTION.
A-000.
     PERFORM B-OPEN.
     PERFORM C-PROCESS.
     PERFORM D-CLOSE.
A-999.
     STOP RUN.

B-OPEN SECTION.
B-000.
     OPEN INPUT IN-FILE.
     OPEN OUTPUT OUT-FILE.
B-999.
     EXIT.

C-PROCESS SECTION.
C-000.
     MOVE SPACES TO IN-REC.
     READ IN-FILE AT END GO TO C-999.
     INSPECT IN-REC REPLACING ALL CHARACTERS
BY UPPER-CASE.
     WRITE OUT-REC FROM IN-REC.
C-110.
     GO TO C-000.
C-999.
     EXIT.

D-CLOSE SECTION.
D-000.
     CLOSE IN-FILE OUT-FILE.
D-999.
     EXIT.
```

COBOL has a "COPY" verb similar to the "LIB" facility in assembler. Using this facility it is possible to create a library of file descriptions and standard paragraphs which can be used to make accurate programs rapidly.

I have only scratched the surface, but I hope this short tour of COBOL helps you understand why it is so popular in commercial applications. Most programs written in COBOL have a long lifetime, and are easy to maintain without specialised knowledge.

COBOL has an advantage over BASIC interpreters in that the data allocation is static, there is no "garbage collection" and programs do not grow beyond their initial sizes. The generated code is therefore more compact and efficient overall.

T. Opyrchal

- - -

Editor's Note: COBOL has been a long time coming for the S50 Bus and the 6BXX. Microware sells a version that runs under OS-9, but this is the first we have seen, that works(!), and runs under FLEX. Again, both groups have a choice.

Bob Nay, who in addition to heading up our COLOR MICRO JOURNAL division, and who also spends a large amount of his time working with the S.E. MEDIA software department, tells me that he is surprised at how pleased buyers of "CRUNCH COBOL" are. Experienced users who have purchased COBOL from S.E. MEDIA have told Bob that it is practically a full **Level one (1)** COBOL.

Not being a COBOL programmer I cannot say as I have spent very little time with S.E. MEDIA CRUNCH COBOL, but from what I hear it is above average for a new product. And for those needing COBOL this should certainly be looked into.

Especially since it has a 50% "SPECIAL" reduction!

Regular Price $199.95

S.E. MEDIA Special: $99.95

See S.E. MEDIA advertising rear cover for FREE phone ordering.

- - -

DEVELOPMENT TERMINAL PROGRAM

CONTINUED FROM LAST MONTH

```
C5F2 52
C5F3 04                 FCB   04
C5F4                EOFMSG
C5F4 4F 55 54 50        FCC   'OUTPUT FILE OPEN ERROR'
C5F8 55 54 20 46
C5FC 49 4C 45 20
C600 4F 50 45 4E
C604 20 45 52 52
C608 4F 52
C60A 04                 FCB   04
C60B                HELPST
C60B 43 20 20 20        FCC   'C - COPY RECORD'
C60F 2D 20 43 4F
C613 50 59 20 52
C617 45 43 4F 52 44
C61A 0D 0A 00           FCB   $D,$A,0
C61D 4E 20 20 20        FCC   /N - DON'T COPY RECORD/
C621 44 4F 4E 27
C625 54 20 43 4F
C629 50 59 20 52
C62D 45 43 4F 52
C631 44
C632 00 0A 00           FCB   $D,$A,0
C635 44 20 2D 20        FCC   'D - DISPLAY DATA RECORD'
C639 44 49 53 50
C63D 4C 41 59 20
C641 44 41 54 41
C645 20 52 45 43
C649 4F 52 44
C64C 00 0A 00           FCB   $D,$A,0
C64F 45 20 2D 20        FCC   'E - EXIT PROGRAM NOW'
C653 45 58 49 54
C657 20 50 52 4F
C65B 47 52 41 4D
C65F 20 4E 4F 57
C663 0D 0A 00           FCB   $D,$A,0
C666 48 20 20 20        FCC   'H - THIS DISPLAY'
C66A 54 48 49 53
C66E 20 44 49 53
C672 50 4C 41 59
C676 04                 FCB   04
                        END   DMPCPY
```

0 ERROR(S) DETECTED

SYMBOL TABLE:

```
ADDBI  CD36   BCNT   C254   DMPCPY C100   BPOINT C10D   BUFPNT C106
CHRLUP C4A0   CLOSE  C404   CMDERR C39C   DATBUF C255   DIDATA C3E8
DISPDA C44C   DISPLP C3C6   DISPLY C40A   DISTRT C439   EADS   C104
EINPUT C5C9   ENDDIS C445   ENDFIL C3FE   ENDGTH C4FA   ENDPTR C30F
ENDSEC C494   EOPMSG C50E   EOPMSP C5F6   ERREAD C515   ERRMSG C401
ENWRITE C546  FCBIN  C040   FCBOUT C111   FLEX   C000   FMS    D406
FMSCLS D403   GETADS C4FB   GETBYT C500   GETCHR C015   GETFIL CD2D
GETREC C4D0   HELPME C3F3   HELPST C60B   HELPUP C47E   LINE   C110
MAINLP C3AB   NOPMPT C106   NXTBYT C4F1   NXTCH  CD27   OPRDER C5A1
OPWRER C3A6   OUTADR CD45   OUTBAT C3E3   OUTHEX CD3C   OUTLUP C45E
PCRLF  CD24   PNTCHR C4B0   PNTPER C4BB   PROMPT C579   PRTCHR C497
PSTRNG CD1E   PUTBYS C530   PUTCHR CD18   PUTREC C51B   RPTERR CD3F
SADS   C252   SDVTEC C55E   SEEFIL C5B1   SEREAD C59A   SETEXT CD33
SEWRIT C5A5   SRECAD C54C   SSTART C549   START C355   STTYPE C3B9
TBCNT  C10F   TEMPX  C107   TERM  C397   TSADS C109   TYPE   C251
WRMS   CD03
```

```
*
*   DMP - Binary disk file dump utility.
*   Displays records of a FLEX binary
*   file, both in hex and in ASCII.
*   For ASCII, the upper bit is removed
*   and the value checked for between
*   hex 20 and 7E.  If it is it is
*   displayed, if not a period is shown.
*   The ASCII string is enclosed in
*   vertical bars.  Each record is headed
*   by its address range and byte count.
*   The values are then displayed is at a
*   time with their start addres on a
*   line followed by the line containing
*   the ASCII.  Each record is shown
*   separatly even if two adjacent records
*   have a contiguous address space.
*   Start blocks are also shown.  To
*   DMP a file, type DMP (filename) at
*   the FLEX prompt.  The extension
*   defaults to ".BIN".
*
*   I place this program in the public
*   domain.  07-NOV-82  J. C. Haugler
*
*   DOS EQUATES - This routine is written
*   using only 6800 opcodes.  Changing
*   the equate below for FLEX from $C000
*   to $A000 should allow it to run on
*   a 6800 as well as a 6809.  At least
*   so they tell me.

C000  FLEX   EQU   $C000
CD03  WARMS  EQU   FLEX+$0B03
CD18  PUTCHR EQU   FLEX+$0B18
CD1E  PSTRNG EQU   FLEX+$0B1E
CD24  PCRLF  EQU   FLEX+$0D24
CD2D  GETFIL EQU   FLEX+$0D2D
CD33  SETEXT EQU   FLEX+$0D33
CD36  ADDBI  EQU   FLEX+$0D36
CD3C  OUTHEX EQU   FLEX+$0D3C
CD3F  RPTERR EQU   FLEX+$0D3F
CD45  OUTADR EQU   FLEX+$0D45
      *
      *  FMS EQUATES
      *
D403  FMSCLS EQU   FLEX+$1403
D406  FMS    EQU   FLEX+$1406
CB40  FCB    EQU   FLEX+$0B40
      *
C100          ORG   FLEX+$0100
      *
      ****PROGRAM START
      *
C100          DMP
C100 20  3D           BRA   START
C102 01       VR      FCB   1
      *
      *  TEMPORARY STORAGE
C103          BPOINT  RMB   2
C105          EADS    RMB   2
C107          SADS    RMB   2
C109          BCNT    RMB   1
C10A          FLAG    RMB   1
C10B          LINE    RMB   1
C10C          BUFR    RMB   19
      *
      *  GET FILENAME AND OPEN FILE
      *
C11F          START
C11F 8E  CB40         LDX   #FCB
C122 BD  CD2D         JSR   GETFIL
C125 25  39           BCS   LINPER
C127 4F               CLRA
C128 BD  CD33         JSR   SETEXT
C12B 86  01           LDAA  #1
C12D A7  84           STAA  0,X
```

```
C12F BD  0406          JSR   FMS
C132 26  2F            BNE   LNOFIL
C134 86  FF            LDAA  #$FF
C136 A7  88 3B         STAA  59,I
                       *
                       ****MAIN LOOP
                       *
C139                   LOOPB
C139 8E  C040          LDI   #FCB
C13C BD  0406          JSR   FMS
C13F 26  1C            BNE   LENDIT
C141 81  02            CMPA  #02
C143 27  24            BEQ   OUTDAT
C145 81  16            CMPA  #16
C147 26  F0            BNE   LOOPB
                       *
                       ****DISPLAY START ADDRESS
                       *
C149 8E  C260          LDI   #SSTART
C14C BD  CD1E          JSR   PSTRNG
C14F BD  C236          JSR   GETADS
C152 8E  C107          LDI   #SADS
C155 BD  CD45          JSR   OUTADR
C158 BD  CD24          JSR   PCRLF
C15B 20  0C            BRA   LOOPB
                       *
                       *  LONG BRANCHES
                       *
C15D 7E  C25E          LENDIT JMP   ENDIT
C160 7E  C254          LINPER JMP   INPERR
C163 7E  C259          LNOFIL JMP   NOFILE
C166 7E  C24F          LRERRD JMP   RERROR
                       *
                       ****DISPLAY BINARY RECORD
                       *
                       *  OUTPUT HEADER DATA
                       *
C169                   OUTDAT
C169 BD  CD24          JSR   PCRLF
C16C BD  C236          JSR   GETADS
C16F 8E  C107          LDI   #SADS
C172 BD  CD45          JSR   OUTADR
C175 86  20            LDAA  #' -
C177 BD  CD18          JSR   PUTCHR

C17A 8E  C040          LDI   #FCB
C17D BD  0406          JSR   FMS
C180 26  E0            BNE   LRERRD
C182 B7  C109          STAA  #CNT
C185 4A                DECA
C186 1F  8B4D          TAB
C189 8E  C107          LDI   #SADS
C18C BD  CD36          JSR   ADDBI
C18F BF  C105          STI   #ADS
C192 8E  C105          LDI   #ADS
C195 BD  CD45          JSR   OUTADR
C198 8E  C28F          LDI   #SBYTE
C19B BD  CD1E          JSR   PSTRNG
C19E 8E  C109          LDI   #CNT
C1A1 BD  CD3C          JSR   OUTHEX
                       *
                       *  MAIN DATA OUTPUT LOOP
                       *
C1A4                   LOOPB
C1A4 BD  CD24          JSR   PCRLF
C1A7 8E  C107          LDI   #SADS
C1AA BD  CD45          JSR   OUTADR
C1AD 86  20            LDAA  #$20
C1AF BD  CD18          JSR   PUTCHR
C1B2 86  10            LDAA  #16
C1B4 B7  C108          STAA  LINC
C1B7 1F  8B4D          TAB
C1BA 8E  C107          LDI   SADS
C1BD BD  CD36          JSR   ADDBI
C1C0 BF  C107          STI   SADS
C1C3 8E  C10C          LDI   #BUFR
C1C6 BF  C103          STI   BPOINT
                       *
                       *  PROCESS INDIVIDUAL BYTES
                       *
C1C9                   LOOPC
C1C9 8E  C040          LDI   #FCB
C1CC BD  0406          JSR   FMS
C1CF 26  95            BNE   LRERRD
C1D1 B7  C10A          STAA  FLAG
C1D4 84  7F            ANDA  #$7F
C1D6 81  20            CMPA  #$20
C1D8 25  04            BLO   PNTPER
C1DA 81  7F            CMPA  #$7F
C1DC 26  02            BNE   PNTCHR
C1DE                   PNTPER
C1DE 86  2E            LDAA  #'.
C1E0                   PNTCHR
C1E0 8E  C103          LDI   BPOINT
C1E3 A7  84            STAA  0,I
C1E5 30  01            INI
C1E7 BF  C103          STI   BPOINT
C1EA 8E  C10A          LDI   #FLAG
C1ED BD  CD3C          JSR   OUTHEX
C1F0 86  20            LDAA  #$20
C1F2 BD  CD18          JSR   PUTCHR
C1F5 7A  C109          DEC   BCNT
C1F8 27  07            BEQ   ENDBLK
C1FA 7A  C108          DEC   LINC

C1FD 26  CA            BNE   LOOPC
C1FF 20  03            BRA   PNTCHR
C201                   ENDBLK
C201 7A  C108          DEC   LINC
                       *
                       *  PUT ASCII DATA LOOP
                       *
C204                   PRTCHR
C204 BD  CD24          JSR   PCRLF
C207 86  10            LDAA  #16
C209 B0  C108          SUBA  LINC
C20C B7  C108          STAA  LINC
C20F 86  7C            LDAA  #$7C
C211 BD  CD18          JSR   PUTCHR
C214 8E  C10C          LDI   #BUFR
C217                   LOOPE
C217 A6  84            LDAA  0,I
C219 30  01            INI
C21B BD  CD1B          JSR   PUTCHR
C21E 7A  C108          DEC   LINC
C221 26  F4            BNE   LOOPE
C223 86  7C            LDAA  #$7C
C225 BD  CD18          JSR   PUTCHR
C228 7D  C109          TST   BCNT
C22B 26  06            BNE   LLOOPD
C22D BD  CD24          JSR   PCRLF
C230 7E  C139          JMP   LOOPB
C233 7E  C1A4          LLOOPD JMP   LOOPD
                       *
                       ****GET ADDRESS SUBROUTINE
                       *
C236                   GETADS
C236 8E  C040          LDI   #FCB
C239 BD  0406          JSR   FMS
C23C 26  0C            BNE   RADRER
C23E B7  C107          STAA  SADS
C241 BD  0406          JSR   FMS
C244 26  06            BNE   RADRER
C246 B7  C108          STAA  SADS+1
C249 39                RTS
                       *
                       ****WRAP UP ROUTINES
                       *
C24A                   RADRER
C24A 8E  C29C          LDI   #SARERR
C24D 20  12            BRA   FIXSTK
C24F                   RERROR
C24F 8E  C2AF          LDI   #SDRERR
C252 20  00            BRA   FIXSTK
C254                   INPERR
C254 8E  C275          LDI   #SINERR
C257 20  08            BRA   FIXSTK
C259                   NOFILE
C259 8E  C281          LDI   #SNOFIL
C25C 20  03            BRA   FIXSTK
```

```
C25E                EMSIT
C25E 8E   C290              LDX   #SEOFIL
C261                FIISTX
C261 BD   CD1E              JSR   PSTRNG
C264 BD   D403              JSR   FMSCLS
C267 BD   CD34              JSR   PCRLF
C26A 7E   C003              JMP   WARMS
                    *
                    ****STRINGS AND THINGS
                    *
C26D                SSTART
C26D 53 54 41 52            FCC   'START:
C271 54 3A 20
C274 04                     FCB   4
C275                SINERR
C275 49 4E 50 55            FCC   'INPUT ERROR'
C279 54 20 45 52
C27D 52 4F 52
C280 04                     FCB   4
C281                SNOFIL
C281 46 49 4C 45            FCC   'FILE NOT FOUND'
C285 20 4E 4F 54
C289 20 46 4F 55
C28D 4E 44
C28F 04                     FCB   4
C290                SEOFIL
C290 45 4E 44 20            FCC   'END OF FILE'
C294 4F 46 20 46
C298 49 4C 45
C29B 04                     FCB   4
C29C                SAREAR
C29C 41 44 44 52            FCC   'ADDRESS READ ERROR'
C2A0 45 53 53 20
C2A4 52 45 41 44
C2A8 20 45 52 52
C2AC 4F 52
C2AE 04                     FCB   4
C2AF                SDAERR
C2AF 44 41 54 41            FCC   'DATA READ ERROR'
C2B3 20 52 45 41
C2B7 44 20 45 52
C2BB 52 4F 52
C2BE 04                     FCB   4
C2BF                SBYTE
C2BF 42 59 54 45            FCC   'BYTE COUNT THIS BLOCK:
C2C3 20 43 4F 55
C2C7 4E 54 20 54
C2CB 48 49 53 20
C2CF 42 4C 4F 43
C2D3 4B 3A 20
C2D6 04                     FCB   4
                            END   ORG

0 ERROR(S) DETECTED

SYMBOL TABLE:

ADDBI  C036   BCNT   C109   BPOINT C103   BUFR   C10C   CMP    C100
EADS   C105   ERDBLK C201   ENDIT  C25E   FCB    CB40   FIISTK C261
FLAG   C10A   FLEI   CDDD   FMS    D40B   FMSCLS D403   GETADS C236
GETFIL CB2D   IMPERR C254   LENBIT C15D   LINC   C10B   LIMPER C160
LLOOPD C233   LNOFIL C163   LOOPB  C139   LOOPC  C1C9   LOOPD  C1A4
LOOPE  C217   LWERRO C166   NOFILE C259   OUTADR CB45   OUTDAT C169
OUTNEI CB3C   PCRLF  CD24   PNTCHR C1E0   PNTPER C10E   PRTCHR C204
PSTRNG CD1E   PUTCHR CD1B   RADRER C24A   REARER C24F   RPTERR CD3F
SADS   C107   SAREAR C29C   SBYTE  C2BF   SDAERR C2AF   SEOFIL C290
SETEXT CB33   SINERR C275   SNOFIL C281   SSTART C26D   START  C11F
VN     C102   WARMS  C003
```

Macintosh-Lisa-Model 16 Users/Reviewers

In the near future we will be running a column for the Macintosh/Lisa and Tandy Model 16 68000 computers. This column will primarily be a 'review' column. As you know there has been a lot of new software, and some hardware, coming fast for all three of these machines. We have been involved in doing some 'contract' reviews for these systems and feel that our readers, as well as readers of our new publication (soon) Pro'View™ need to know what actually does work as advertised and the indepth reporting of our findings.

Pro'View will be a newsprint tabloid devoted primarily to reviews for these three machines. However, there will be the usual advertising and other support articles that make such a publication desirable to the owner/user. Also there will be how-to articles, about the 'guts' of the system that no other magazine now covers.

We have developed a rather excellent reputation for getting to the 'insides' of computers and telling our readers and clients those things that are very important to most users. I wish I could tell you the names of the large computer companies and others that have come to us for assistance. However, as you well know they don't want their competitors to know their trends of interest, so mum is the word. But I can tell you that as a result, you have a better selection of both hardware and software, than if we were not involved. For the relatively small amount of S50 Bus computers as opposed to the 'other side' we have a wealth of excellent soft/hardware. Fact is, after reviewing a lot of stuff for other systems, FLEX™, STAR-DOS™, UniFLEX™ and OS-9™ are FAR - **FAR** - superior to most all the other disk systems. Our problem has been applications software, or not enough of it or the right kind or support, however, that is another subject.

Back to the original subject, our new column and Pro'View. What we now need is reviewers. Owners or users of one or more of the subject systems. Authors who can properly dissect a piece of software and report objectively and honestly the results. What advantages or disadvantages to the end user, why he/she should or should not consider purchase of subject item, ease of use, documentation and all those other attributes that are essential and vital pieces of knowledge a prospective purchaser should have at hand when the time comes to make a decision.

The reviewer will be compensated in one of several ways. First, some will be paid outright, in case the item is to be returned after review, others will get to keep the item (most seem to prefer keeping the item) and some will be done in our lab, in which case it does not matter.

Now, if you are interested and qualified PLEASE drop me a line or call me on the S.E. MEDIA WATTS line 1-800-338-6800, tell the operator that you want to talk to Don, Bob or Larry about doing 68000 reviews for the subject systems mentioned above.

We have found a few good items already and a lot of junk, we need to let our readers know what we found, good or bad. So let me hear from you soon. Also, nearly forgot; it will probably make you a world renown author. How about that?

DMW

- - -

DO

A command file processor for FLEX-09

by Ben Staghekke
Twickelerlaan 8
7495 VG Ambt-Delden
the Netherlands.

1. Summary

DO is a utility that processes commands from a FLEX textfile. Its enhancements over FLEX's EXEC are:
- DO allows the use of parameters.
- Programs that run from a DO-file, can read their input from either the terminal or the command file.
- In the command file, you can specify if an error stops the execution of the file or not.
- You can abort the execution of a command file in a decent way.

- You can insert comments in the command file.
- DO echoes all that it reads from the command file, to the terminal.

2. Command format

A command file is invoked by typing

+++DO fileneme [param1,param2,...,param n]

If you don't specify a drive number, DO assumes that the command file resides on the system disk. The default extension is ".MIC". Parameters can consist of any text; they are separated by commas. To insert a comma in a parameter, precede it by a "@". An example: x@,y is one parameter, with the value x,y. A @ is passed by entering it twice, e.g. x@@y is 'expanded' into x@y. You can give empty parameters by entering 2 consecutive commas, e.g. x,,y are 3 parameters, of which the second is empty. At most 10 parameters are allowed, and together, they should not take more than 128 characters. To abort a command file while it is running, depress control-A several times. After finishing the current command, DO stops.

3. Command file format

3.1 An example

Using BUILD or EDIT, create the following text file:

```
! 'A is the label, 'B is the disk number.
+NEWDISK 1
Y'A
'B
! type Y or N
+CAT 1
! Finished...
```

Call this file ND.MIC, and put it on the system disk. Now put a scratch disk in drive 1 and type

+++DO ND MYDISK,12

You will see the following:

```
! this file formats the disk in drive 1.
! MYDISK is the label, 12 is the disk number.
++NEWDISK 1
SCRATCH DISK IN DRIVE 1? Y
VOLUME NAME: MYDISK
VOLUME NUMBER: 12
ARE YOU SURE? ! type Y or N
```

At this moment, the execution of the command file stops. Type a "Y". Now Newdisk starts running, and when it is done, the (empty) disk catalog appears.

3.2 Special characters

Usually, "DO" sends the text from the command file to either FLEX or the running program. However, a few characters have a special meaning:

+ A plus sign, when it appears at the begin of a line, indicates to DO that the following line is a command.

Example:

```
+CAT 1
```

& The ampersand, when it appears right after the afore-mentioned '+', indicates that, if the command gives an error, DO should not abort.
Example:

```
+&DELETE 1.GARBAGE.TMP
```

If the file GARBAGE.TMP doesn't exist, Delete prints an error message, but the command file proceeds. +DELETE 1.GARBAGE.TMP Now, if the file doesn't exist, the following is printed:

```
THE SPECIFIED FILE DOESN'T EXIST

...Command file aborted

+++
```

! The exclamation mark introduces a comment. Anything that follows it, up to the next return, is only typed on the terminal.

Example:

```
! this is a comment line
+CAT 1   ! The CAT is done; the rest is comment, too.
```

'The single quote, followed by a letter A..J (or a..j) invokes parameter substitution from the command line. 'A is the first parameter from the line, 'B the second, etcetera. Note that you can't have more than 10 parameters. Note also, that 'A and 'a mean the same parameter. Example: a command line +++DO KILIT GARBAGE.TMP,! The file KILIT.MIC contains:

```
+DELETE 'A
YY
+'bCAT 1
```

It is expanded into:

```
++DELETE GARBAGE.TMP
DELETE "1.GARBAGE.TMP"? Y
ARE YOU SURE? Y
++!CAT 1
```

In this example, 'A is expanded into GARBAGE.TMP, and 'b into a "!". This exclamation mark changes the command line into a comment, and therefore the CAT command is not done.

@ The commercial 'at' makes the next character loose its special meaning. This also holds for the "@" itself. Examples: @'AB expands into 'AB

```
@! This is NOT a comment line..
```

@@Z expands into @z

3.3 Special features

If DO finds 'unused' input lines between commands in a file, it ignores them. On the other hand, if not enough input is available from the file, DO takes the input from the console keyboard. Example:

```
+CAT 1
This line is not used
+DELETE GARBAGE.TMP
+CAT 1
```

Line 2 is not used for input, and therefore ignored. The DELETE command asks 2 questions. Because there are no answers for these questions in the command file, DO opens the terminal for input.

While DO is running, the TTYSET PauSe feature is switched off. This way, DO can happily munch through a command file on its own, without you having to sit there and hit the escape key all the time. When DO exits, the old pause condition is restored.

4. Chaining DO command files.

A DO command file may contain a call of another DO file. The new command file has a new set of parameters, and there is no way to access the old set.

Further, there is no return to the previous command file, once the new file is finished. DO doesn't mind if a command file calls itself. Example: file 0.GROW.MIC

```
   ! this line grows 'Abigger
   ! until it becomes too long or
   ! it is aborted with control-A.
   +DO GROW bigger and 'A
```

Start the file like this:

```
+++DO GROW
```

and watch it grow...

5. Memory requirements

DO loads into 2 separate parts of the memory. One part loads into the utility command space; it is overwritten once DO starts running. The other part loads at $BC00, at the end of the user memory. DO changes MEMEND into $BBFF when it starts, and restores it when it exits. DO prints an error message if, upon entry, MEMEND is not equal to $BFFF.

6. Side effects

Because DO changes the INCHR vector, don't use the I utility in a DO file. (obviously, there is no need to do so). There is no objection against using the O utility. Because, upon exit, DO restores its version of the TTYSET PS flag, a TTYSET PS=Y/N command inside a DO file doesn't work.

7. Getting DO to run

To get DO running, you need DO's source. Either spend a few nights typing, or get hold of the source another way. Maybe 68 Micro Journal can organise something? There is nothing fancy about assembling DO. It was written for Flex's ASMB. Call the generated object DO.CMD, and put it on the system drive. That's all!

8. Adapting DO

A few things in DO's source can be changed easily:
the allowed number of parameters (currently 10).
the size of the command buffer (currently 128 chars).
the break character (currently control-A).
the area where the run-time code will be.
Note that only if this parameter = $BC00, the code that checks and modifies MEMEND, is assembled. This is because in my system, I load the runtime-code at $E000, so I don't have to mess around with MEMEND.
the default extension for the command files (currently .MIC).

The first 4 items are defined by equates, placed together at the front of the source. The default extension is defined immediately after the first ORG statement.

9. Conclusion

The command syntax and some features of DO were copied from the DO command that is used on the DECsystem-20 computer. I don't know who wrote the DEC version. I hope you will enjoy using DO and I am very interested in any comments on it.

\- - -

Editor's Note: The source listing will be available on disk with all text files - AS PUBLISHED - for $29.95. This includes the cost of the disk (specify 5 or 8 inch), all postage and handling charges also. Also included will be the program - AS PUBLISHED - assembled into a DO.CMD file.

\- - -

```
* OPT
        TTL     DO: Flex command file processor
*
*****************************************************************
*                                                               *
*  D O                                                          *
*  ===                                                          *
*   FLEX COMMAND FILE PROCESSOR                                 *
*                                                               *
*   DO is a utility to process a file as a list of              *
*   Flex commands. Do is much more powerful than EXEC,          *
*   because it can handle parameters.                           *
*                                                               *
*   Command syntax:                                             *
* +++DO <filespec> [param,param,param...]                        *
*                                                               *
*   A Flex command file has a default extension '.mic'          *
*   and is located on the system drive.                         *
*   A line from a command file is only executed if it           *
*   is preceded by a '+'.                                       *
*   An input line for a utility does not start with any         *
*   special character, and everything that is read              *
*   between '!' and the following <CR> is treated as            *
*   comment, i.e. it is not interpreted, only printed.          *
*   If a utility asks for more input than is available,         *
*   you can enter it from the keyboard.                         *
*   From within lines in the command file, you can              *
*   refer to parameters, given with the 'DO' command,           *
*   using a single quote ('), followed by a letter A..J         *
*   (10 parameters allowed).                                    *
*   To pass a literal character (like a , or a '), pre-         *
*   cede it by a '@'.                                           *
*   To prevent DO from aborting if an error occurs in           *
*   a command, precede the command name by a '-', like:         *
*   +-DELETE FRUT.TMP                                           *
*   To stop the execution of a micfile, type ^A. After          *
*   the current command is finished, DO stops.                  *
*   From within a micfile, you can activate a next one,         *
*   but the activation acts as a jump, not as a                 *
*   subroutine call.                                            *
*                                                               *
*   DATE:    1 Feb 1983                                         *
*   Written by                                                  *
*      Ben Staghekke,                                           *
*      Twickelerlaan 8,                                         *
*      7495 VG  Ambt-Delden                                     *
*      the Netherlands                                          *
*                                                               *
*****************************************************************
*
0003 VM      EQU   3          Date 84-04-13


*
* Customizable constants
*
000A MXPARM  EQU   10         nbr of params allowed
0080 BUFLEN  EQU   128        length area for parameters
0001 BRKCHR  EQU   $1         break char (control-A)
BC00 CODARE  EQU   $BC00      runtime code area

* NOTE
* CODARE being $BC00 causes the code to be assembled,
* that checks and lowers MEMEND during DO's life.


* Fixed constants

000D CR      EQU   $D         CHAR CARRIAGE RETURN
000A LF      EQU   $A         CHAR LINEFEED


*
* FLEX references
*

* routines in DOS

CD03 WARMS   EQU   $CD03      Dos warm start entry
CD09 INCH    EQU   $CD09      input char vector
CD0C INCH2   EQU   $CD0C      idem, always from console
```

```
CD18  PUTCHR  EQU   $CD18     output a char
CD1E  PSTRNG  EQU   $CD1E     print a string
CD24  PCRLF   EQU   $CD24     print a newline
CD27  NXTCH   EQU   $CD27     get next character from buffer
CD2D  GETFIL  EQU   $CD2D     process a filespec
CD3F  APTERR  EQU   $CD3F     report an error
CD48  DOCMND  EQU   $CD48     do command
CD4E  TSTA    EQU   $CD4E     rtn <NE> if char typed

      * variables in DOS

CC09  TTPAUS  EQU   $CC09     ttyset pause flag
CC0D  DEFDRV  EQU   $CC0D     default drive getfil
CC14  LBFPTR  EQU   $CC14     line buffer pointer
CC1A  CURLNR  EQU   $CC1A     current line number
CC28  DCMFLG  EQU   $CC28     Do-command flag
CC2B  MEMEND  EQU   $CC2B     memory end address

      * FMS routines

D403  FMSCLS  EQU   $D403     close FMS
D406  FMS     EQU   $D406     FMS call

      * FMS instruction codes

0001  FMOPER  EQU   1         open for read
0004  FMCLOS  EQU   4         close FCB

      * FMS error codes

0008  FPAEOF  EQU   8         read past end of file

      * layout of an FCB

0000  FCBFCD  EQU   0         function code
0001  FCBERR  EQU   1         error status byte
0002  FCBACT  EQU   2         activity status
000C  FCBEXT  EQU   12        file extension
0140  FCBLEN  EQU   320       length FCB in bytes
                    *
                    * This is the TRANSIENT PART of DO.
                    * This part initializes the reading
                    * of the command file, and reads
                    * the command line.
                    * This code is overwritten once the command
                    * line of the DO command has been interpreted.
                    * This part is NOT position-independant.
                    *
                    * Because lots of FLEX commands expect the DPR to
                    * contain 0, the DPR is not used in DO.
                    *
                          SETDP
C100                      ORG   $C100     utility command space
C100 20  04               BRA   START
C102 03                   FCB   VN        CURRENT VERSION

C103 4D 49 43       DEFEXT FCC    'MIC'   default extension command file

           C106     START  EQU   *
                    *
                    * Find out where the lower part is loaded
                    *
C106 CE  BC00             LDU   #BASE     base register for local variables

C109 10EF C9 009E         STS   SAVSTK-BASE,U save stack

C10E 7D  CC28             TST   DCMFLG    if nested do
C111 26  12               BNE   DOITIO    then skip

C113 8E  CC2B             LDX   #MEMEND
C116 BF  BCA0             STX   SVMEND    KEEP MEMEND

                    * check if MEMEND OK

C119 8D  5E               BSR   CKMEME

                    * Disable pause feature

C11B B6  CC09             LDA   TTPAUS    TTYSET PAUSE FLAG

C11E A7  C9 009B          STA   PSCNO-BASE,U KEEP OLD VALUE
C122 7F  CC09             CLR   TTPAUS    PAUSE DISABLED

           C125     DOITIO EQU   *         INIT ALWAYS
C125 6F  C9 0001          CLR   IMPARM-BASE,U
C129 86  0D               LDA   #CR
C12B A7  C9 009A          STA   LSTCHR-BASE,U INIT VARS

                    * read file spec from command line

C12F 7C  CC0D             INC   DEFDRV    default drive is system drive
C132 10  C9 00A2          LEAX  MWFCB-BASE,U
C136 BD  CD2D             JSR   GETFIL    GET FILE SPEC
C139 24  04               BCC   SPECOK

           C13B     DKER1  EQU   *
C13B 6E  C9 0C9C          JMP   DKERR-BASE,U REPORT DISK ERROR

           C13F     SPECOK EQU   *
C13F 6D  0C               TST   FCBEXT,X  IF USER SPECIFIED EXTENSION
C141 26  0C               BNE   XTCOMP    THEN COMPLETE
C143 86  C103             LDA   DEFEXT    else setup default
C146 10BE C104            LDY   DEFEXT+1
C14A A7  0C               STA   FCBEXT,X
C14C 10AF 0D              STY   FCBEXT+1,X
           C14F     XTCOMP EQU   *         EXTENSION COMPLETE

                    * Open file and close it again, to keep it
                    * out of FCB chain.

C14F 86  01               LDA   #FMOPER   OPEN FILE FOR READ
C151 A7  84               STA   FCBFCD,X  INTO FCB
C153 BD  0406             JSR   FMS       DO OPEN
C156 26  E3               BNE   DKER1     OPEN FAILED


C158 86  04               LDA   #FMCLOS
C15A A7  84               STA   1
C15C BD  0406             JSR   FMS       CLOSE FILE AGAIN
C15F 26  DA               BNE   DKER1     DISK ERROR IF WRONG

C161 86  01               LDA   #1        FILE OPEN FOR READ
C163 A7  02               STA   FCBACT,X  IN ACTIVITY STATUS

C165 6F  84               CLR   FCBFCD,X  FUNCTION READ NEXT BYTE

                    * Read parameters from command line

C167 BD  C1AC             JSR   RDCMDL

                    * If a nested DO, jump to warms, otherwise
                    * jump at command loop

C16A 7D  CC28             TST   DCMFLG
C16D 27  03               BEQ   SOLDMP
C16F 7E  C003             JMP   WARMS

           C172     SOLDMP EQU   *
C172 7C  CC28             INC   DCMFLG    NOW WE ARE IN DO
C175 6E  C9 01E2          JMP   LOPART-BASE,U IN OTHER PART OF MEM


                    *
                    * Proc CKMEME checks if MEMEND equals $BFFF.
                    * If it isn't, the proc prints an error message
                    * and forces DO to exit.
                    *
           C179     CKMEME EQU   *
C179 BE  CC2B             LDX   MEMEND    If MEMEND has correct value then
C17C 8C  BFFF             CMPX  #$BFFF
C17F 26  07               BNE   CKME10

C181 8E  BBFF             LDX   #CODAPE-1
C184 BF  CC2B             STX   MEMEND    install lower memend
C187 39                   RTS             and exit

           C188     CKME10 EQU   *         else: wrong memend value
C188 30  8D 0004          LEAX  MEMERR,PCR point at string
C18C 6E  C9 021A          JMP   PRVERR-BASE,U print it and exit
```

```
                    C190  MEMERR  EQU   *
C190 53 6F 72 72            FCC   'Sorry; your MEMEND is wrong',4
C194 79 3B 20 79
C198 6F 75 72 20
C19C 4D 45 4D 45
C1A0 4E 44 20 69
C1A4 73 20 77 72
C1A8 6F 6E 67 04

                    *
                    * Proc RD-CMD-LINE reads the command line into
                    * a private buffer. The filename must have been
                    * eaten out beforehand.
                    *
                    C1AC  RDCMDL  EQU   *
C1AC 8D   C1E7              JSR   INITCB  INIT COMMAND BUFFER
                    C1AF  RDCM10  EQU   *
C1AF 8D   23                BSR   NXTCHL  NEXT CHAR FROM CMD LINE
C1B1 81   00                CMPA  #CR
C1B3 27   1B                BEQ   RDCMEI  IF NOT NEWLINE THEN
C1B5 8D   7D                BSR   NXPARM  A NEW PARAM HAS STARTED

                    C1B7  RDCM20  EQU   *
C1B7 A0   C9 0000           FST   LITERA-BASE,U
C1BB 26   04                BNE   RDCM30  TREAT LITERAL AS NORMAL CHAR
C1BD B1   2C                CMPA  #',     WHILE NOT PARAM SEPARATOR DO
C1BF 27   04                BEQ   RDCM40

                    C1C1  RDCM30  EQU   *
C1C1 8D   47                BSR   PTCPAR  PUT CHAR IN PARM BUFFER
C1C3 8D   0F                BSR   NXTCHL  NEXT CHAR FROM COMMAND LINE
C1C5 81   0D                CMPA  #CR
C1C7 27   07                BEQ   RDCMEI  IF NEWLINE THEN QUIT
C1C9 20   EC                BRA   RDCM20  REP;

                    C1CB  RDCM40  EQU   *         PARAMS SEPARATOR FOUND
C1CB 4F                     CLRA
C1CC 8D   3C                BSR   PTCPAR  PUT 'END OF PARM'
C1CE 20   BF                BRA   RDCM10  REP;

                    C1D0  RDCMEI  EQU   *
C1D0 4F                     CLRA
C1D1 8D   37                BSR   PTCPAR  TERMINATE LAST PARM
C1D3 39                     RTS

                    *
                    * Proc NXTCHL reads the next char from
                    * the command line, processing literals
                    *
                    C1D4  NXTCHL  EQU   *
C1D4 6F   C9 0000           CLR   LITERA-BASE,U PRESET: CHAR NOT LITERAL
C1D8 8D   CB27              JSR   NXTCH   GET NEXT CHAR
C1DB 81   40                CMPA  #'@     IF LITERAL MARKER THEN
C1DD 26   07                BNE   NXTCH1
C1DF 6C   C9 0000           INC   LITERA-BASE,U SET FLAG
C1E3 8D   CB27              JSR   NXTCH   GET LITERAL CHAR
                    C1E6  NXTCH1  EQU   *
C1E6 39                     RTS

                    *
                    * INITCB initializes the command buffer
                    *
                    C1E7  INITCB  EQU   *
C1E7 30   C9 0081           LEAX  BUFFER+BUFLEN-1-BASE,U
C1EB 6F   84                CLR   ,X
C1ED 1F   10                TFR   X,D
C1EF 30   C9 0082           LEAX  POINTER-BASE,U
C1F3 AF   C9 0096           STX   CURPOI-BASE,U

C1F7 31   C9 0096           LEAY  2+NXPARM+POINTER-BASE,U
C1FB 34   20                PSHS  Y       END OF PARAM POINTERS
                    C1FD  IN10    EQU   *
C1FD E8   B1                STB   ,X++    FILL ALL PARM POINTERS WITH OWNY
C1FF AC   E4                CMPX  ,S
C201 26   FA                BNE   IN10
C203 32   62                LEAS  2,S     DROP END VALUE

C205 30   C9 0002           LEAX  BUFFER-BASE,U X -> PARM BUFFER
C209 39                     RTS
```

```
                    *
                    * Proc PTCPAR puts A into parm buffer
                    * AT [X], X := X + 1
                    *
                    C20A  PTCPAR  EQU   *
C20A 31   C9 0081           LEAY  BUFFER+BUFLEN-1-BASE,U END ADDRESS OF BUFFER
C20E 34   20                PSHS  Y
C210 AC   E1                CMPX  ,S++    IF END ADDRESS
C212 26   07                BNE   PTCP10
C214 BE   C21E              LDX   #PTCPER THEN REPORT ERROR
C217 6E   C9 021A           JMP   PRVERR-BASE,U

                    C21B  PTCP10  EQU   *
C21B A7   80                STA   X+
C21D 39                     RTS

C21E 3F 20 50 61  PTCPER  FCC   '? Parameters too long'
C222 72 61 6D 65
C226 74 65 72 73
C22A 20 74 6F 6F
C22E 20 6C 6F 6E
C232 67
C233 04                     FCB   4

                    *
                    * Proc NXPARM puts X into the next parm pointer
                    *
                    C234  NXPARM  EQU   *
C234 34   16                PSHS  D,X
C236 1F   10                TFR   X,D

C238 AE   C9 0096           LDX   CURPOI-BASE,U
C23C 3E   C9 0096           LEAY  2+NXPARM+POINTER-BASE,U END OF BUFFER
C240 34   20                PSHS  Y
C242 AC   E1                CMPX  ,S++    IF BUFFER FULL THEN
C244 26   06                BNE   NXPA10
C246 9E   C254              LDX   #NXERPM REPORT ERROR
C249 7E   BE1A              JMP   PRVERR

                    C24C  NXPA10  EQU   *
C24C E8   01                STB   ,X++    PUT POINTER
C24E AF   C9 0096           STX   CURPOI-BASE,U PRESET NEXT POINTER ADDRESS

C252 35   96                PULS  D,X,PC

C254 3F 20 54 6F  NXERPM  FCC   '? Too many parameters'
C258 6F 20 6D 61
C25C 6E 79 20 70
C260 61 72 61 6D
C264 65 74 65 72
C268 73
C269 04                     FCB   4

                    *
                    * The next part is located at 'codare'.
                    *
                    * Note that this part is fully position-
                    * independant. However, the offset between
                    * the variables and the following code is
                    * supposed to be fixed.
                    *
                    * VARIABLES
                    * To prevent trouble at load-time, use a
                    * FCB pseudo for the first variable.
                    *
BC00                        ORG   CODARE
                    BC00  BASE    EQU   *       base address variables and code

BC00 00             LITERA  FCB   0       CURRENT CHAR IS LITERAL
BC01 00             INPARM  FCB   0       # 0 IF EXPANDING A PARAMETER
BC02                BUFFER  RMB   BUFLEN  PARAMETER BUFFER
BC82                POINTER RMB   2*NXPARM PARAMETER POINTERS INTO BUFFER
BC96                CURPOI  RMB   2       CURRENT POINTER
BC98                CMPTR   RMB   2       COMMAND BUFFER POINTER
BC9A 0D             LSTCHR  FCB   CR      LAST CHAR READ
BC9B 00             PLUSFLG FCB   0       PLUS READ
BC9C 00             CAFTERR FCB   0       CONTINUE AFTER ERROR
BC9D                PSCMD   RMB   1       SAVED TTYSET PS CONDITION
BC9E                SAVSTK  RMB   2       SAVED STACK POINTER
BCA0                SVMEMD  RMB   2       SAVED MEMEND VALUE
```

```
                    *
                    * Local FCB used for reading the file
                    *
BCA2                MYFCB   RMB     FCBLEN

                    *
                    * Main part of file processor
                    *

            BDE2    LOPART  EQU     *
BDE2 BD   C024              JSR     PCRLF       PRINT NEWLINE

            BDE5    WITLIN  EQU     *           WHILE NOT END OF FILE DO

BDE5 BD   CD4E              JSR     TRWSTA      IF NO CHAR TYPED
BDE8 27   02                BEQ     WITL10      THEN NEXT LINE
BDEA 80   60                BSR     KBDIN       : TO CHECK FOR ABORT

BDEC 10EE C9 009E   WITL10  LDS     SAVSTK-BASE,U
BDF1 BD   0403              JSR     FMSCLS      CLOSE FMS
BDF4 17   0087              LBSR    RDLINE      READ NEXT LINE FROM FILE

BDF7 30   8D 011A           LEAX    PARAM,PCR PARAM INPUT CHAR ROUTINE
BDFB BF   CD0A              STX     INCH+1      INTO INPUT CHAR VECTOR
BDFE BD   CB4B              JSR     DOCMND      PROCESS THIS FILE LINE
BE01 6D   C9 009C           TST     CAFTERR-BASE,U
BE05 26   0E                BNE     WITLIN      DON'T CHECK FOR ERRORS

BE07 5D                     TSTB
BE08 27   DB                BEQ     WITLIN      NEXT LINE IF OK
BE0A 20   16                BRA     PRV10       PRINT "...ABORTED"

                    * Process a disk error

            BE0C    DkERR   EQU     *
BE0C 10EE C9 009E           LDS     SAVSTK-BASE,U RESTORE STACK
BE11 30   C9 00A2           LEAX    MYFCB-BASE,U POINT AT FCB
BE15 BD   CB3F              JSR     RPTERR      REPORT ERROR
BE19 20   08                BRA     PRV10

                    * Process private error

            BE1A    PRVERR  EQU     *           X -> ERROR TEXT
BE1A 10EE C9 009E           LDS     SAVSTK-BASE,U
BE1F BD   CD1E              JSR     PSTRNG      WRITE TEXT
            0BE2    PRV10   EQU     *
BE22 10EE C9 009E           LDS     SAVSTK-BASE,U RESTORE STACK
BE27 30   8D 003D           LEAX    ABTERR,PCR
BE2B BD   CD1E              JSR     PSTRNG      PRINT "...ABORTED"

                    * Exit

            BE2E    QUIT    EQU     *
BE2E 10EE C9 009E           LDS     SAVSTK-BASE,U

BE33 BD   0403              JSR     FMSCLS      CLOSE FMS

BE36 7F   CC1A              CLR     CURLNR      CURRENT LINE := 0

BE39 A6   C9 0090           LDA     PSCND-BASE,U RESTORE PAUSE FLAG,
BE3D B7   CC09              STA     TTPAUS
BE40 7F   CC2B              CLR     OCMFLG

BE43 8E   BCA0              LDX     SVMEMD      RESTORE MEMEND
BE46 BF   CC2D              STX     MEMEND
BE49 7E   CD03              JMP     WARMB       AND EXIT

                    *
                    * KBDIN reads a char from the keyboard.
                    * If it is a break character,
                    * then DO aborts.
                    *
BE4C BD   CD0C      KBDIN   JSR     INCH2       get the char
BE4F B1   01                CMPA    #BRKCHR     check if a break char
BE51 27   01                BEQ     KB10
BE53 39                     RTS

            BE54    KB10    EQU     *           it's a break
BE54 30   8D 0002           LEAX    BREAK,PCR pickup break text
BE58 20   C0                BRA     PRVERR

            BE5A    BREAK   EQU     *
BE5A 42 20 52 20            FCC     'B R E A K :'
BE5E 45 20 41 20
BE62 4B 20 3A
BE65 04                     FCB     4

            BE66    ABTERR  EQU     *
BE66 2E 2E 2E 63            FCC     '...command file aborted'
BE6A 6F 6D 6D 61
BE6E 6E 64 20 66
BE72 69 6C 65 20
BE76 61 62 6F 72
BE7A 74 65 64
BE7D 04                     FCB     4

                    *
                    * Procedure RDLINE reads the next line
                    * from the file and puts it in the line
                    * buffer
                    *
            BE7E    RDLINE  EQU     *

BE7E 8E   C080              LDX     #$C080      KEYBOARD BUFFER ADDRESS
BE81 BF   CC14              STX     LBFPTR      INTO LINE BUFFER POINTER
BE84 AF   C9 0098           STX     CBFPTR-BASE,U INIT LOCAL POINTER
BE88 6D   C9 0098           TST     PLUSFLG-BASE,U
BE8C 27   0E                BEQ     RDL10

BE8E 6F   C9 0098           CLR     PLUSFLG-BASE,U
BE92 BD   CD24              JSR     PCRLF
BE95 86   2B                LDA     #'+
BE97 BD   CD1B              JSR     PUTCHR
BE9A 20   2E                BRA     WITCHR

                    * Eat chars until end ling starts

            BE9C    RDL10   EQU     *
BE9C A6   C9 009A           LDA     LSTCHR-BASE,U
BEA0 81   0D                CMPA    #CR         IF LAST CHAR NOT NEWLINE
BEA2 27   08                BEQ     RDL20
BEA4 BD   CD50              BSR     ECHO        THEN GET NEXT CHAR
BEA6 81   02                CMPA    #2          IF START OF BINARY THEN
BEA8 27   31                BEQ     BINFIL      TELL USER
BEAA 20   F0                BRA     RDL10       AND AGAIN

            BEAC    RDL20   EQU     *
BEAC BD   CD24              JSR     PCRLF       PRINT NEWLINE
BEAF BD   45                BSR     ECHO        GET NEXT CHAR
BEB1 81   02                CMPA    #2          IF START OF BINARY THEN
BEB3 27   26                BEQ     BINFIL      TELL USER

BEB5 81   2B                CMPA    #'+         IF NOT A + THEN
BEB7 26   E3                BNE     RDL10       EAT CHARS AGAIN
BEB9 BD   CD1B              JSR     PUTCHR
BEBC 90   38                BSR     ECHO

BEBE 6F   C9 009C           CLR     CAFTERR-BASE,U PRESET
BEC2 81   26                CMPA    #'&         IF CONTINUE AFTER ERROR MARKER
BEC4 26   06                BNE     WITC10
BEC6 6C   C9 009C           INC     CAFTERR-BASE,U THEN SET FLAG

            BECA    WITCHR  EQU     *           NEXT CHARACTER
BECA BD   2A                BSR     ECHO        GET ONE

            BECC    WITC10  EQU     *

BECC AE   C9 0098           LDX     CBFPTR-BASE,U GET LOCAL BUFFER POINTER
BED0 A7   80                STA     ,X+         PUT CHAR
BED2 AF   C9 0098           STX     CBFPTR-BASE,U UPDATE POINTER

BED6 81   0D                CMPA    #$0D        IF CHAR = RETURN
BED8 26   F0                BNE     WITCHR      THEN NEXT CHAR
BEDA 39                     RTS

            BEDB    BINFIL  EQU     *           BINARY FILE OPENED
BEDB 30   8D 0003           LEAX    ILLFIL,PCR
BEDF 16   FF3B              LBRA    PRVERR      GO REPORT

            BEE2    ILLFIL  EQU     *
BEE2 3F 20 49 6C            FCC     '? Illegal file type'
```

```
BEE6 6C 65 67 61
BEEA 6C 20 66 69
BEEE 6C 65 20 74
BEF2 79 70 65
BEF5 04                 FCB   4

                        *
                        * Proc ECHO reads the next char from MITIN and
                        * echoes it to the out stream.
                        * Comment is echoed, but stripped
                        * B is kept
                        *
          BEF6 ECHO     EQU   *
BEF6 34   04            PSHS  B
          BEF8 ECHO10   EQU   *
BEF8 8D   4A            BSR   MITIN     GET NEXT IN CHAR

BEFA 6D   C4            TST   LITERA-BASE,U IF LITERAL
BEFC 26   06            BNE   ECHO20    THEN READY
BEFE 81   2A            CMPA  #'*       IF NOT COMMENT INTRODUCER
BF00 26   02            BNE   ECHO20    THEN READY
BF02 8D   07            BSR   DOCOMM    EAT COMMENT

          BF04 ECHO20   EQU   *         CHAR OK
BF04 34   02            PSHS  A         KEEP IT
BF06 BD   CD1B          JSR   PUTCHR    PRINT IT
BF09 35   86            PULS  D,PC      DONE

                        * DOCOMM eats comment

          BF0B DOCOMM   EQU   *
BF0B BD   CD1B          JSR   PUTCHR    PRINT CHAR
BF0E 8D   34            BSR   MITIN     GET NEXT CHAR
BF10 81   0D            CMPA  #CR       IF NOT CR THEN
BF12 26   F7            BNE   DOCOMM    REPEAT

BF14 39                 RTS
                        *
                        * Proc PARAM1 reads a char via ECHO. If it is
                        * the first char of a new command line then
                        * a flag is set and kbdinput is opened
                        *
          BF15 PARAM1   EQU   *
BF15 34   04            PSHS  B
BF17 E6   C9 009A       LDB   LSTCHR-BASE,U GET LAST CHAR READ

BF1B 6D   C9 009B       TST   PLUSFLG-BASE,U IF KBD INPUT ACTIVE
BF1F 26   17            BNE   PARAM2    THEN USE IT

BF21 8D   03            BSR   ECHO      GET NEW CHAR

BF23 6D   C4            TST   LITERA-BASE,U IF LITERAL
BF25 26   1B            BNE   PARAM3    THEN OK
BF27 81   2B            CMPA  #'+       IF NOT +
BF29 26   17            BNE   PARAM3    THEN OK
BF2B C1   0D            CMPB  #CR       IF PREVIOUS CHAR # CR
BF2D 26   13            BNE   PARAM3    THEN OK

BF2F 86   08      PARAM1 LDA    #8        BACKSPACE
BF31 8D   CD1B          JSR   PUTCHR
BF34 6C   C9 009B       INC   PLUSFLG-BASE,U

BF38 17   FF11    PARAM2 LBSR   KBDIN    get char from keyboard
BF3B 81   0D            CMPA  #CR
BF3D 26   03            BNE   PARAM3    NONE
BF3F 8D   CD1B          JSR   PUTCHR

BF42 35   04      PARAM3 PULS   B,PC      A CONTAINS CHAR
                        *
                        * Proc MITIN reads the next char, either from
                        * the infile, or from an expanded param
                        *
          BF44 MITIN    EQU   *
BF44 34   10            PSHS  X
BF46 6D   41            TST   EXPARM-BASE,U IF NOT EXPANDING A PARM THEN
BF48 26   33            BNE   MITFPA
BF4A 8D   76            BSR   MIFMS     GET NEXT CHAR

BF4C 6D   C4            TST   LITERA-BASE,U IF LITERAL
BF4E 26   04            BNE   MIT05     THEN EXIT

BF50 81   27            CMPA  #''       IF NOT PARAM CALL
BF52 27   02            BEQ   MIT10
BF54 35   90      MIT05 PULS  X,PC      THEN EXIT

          BF56 MIT10    EQU   *         ELSE (EXPAND A PARM)
BF56 8D   6A            BSR   MIFMS     FIND PARAM NUMBER
BF58 84   5F            ANDA  #$5F      CONVERT TO UPPER
BF5A 80   41            SUBA  #'A
BF5C 25   04            BCS   WRONG     TOO LOW
BF5E 81   0A            CMPA  #MAXPARM
BF60 25   07            BCS   MIT15     OK TOO HIGH:
          BF62 WRONG    EQU   *
BF62 30   8D 002B       LEAX  MITILR,PCR 'ILLEGAL REF'
BF66 16   FE81          LBRA  PRVERR

          BF69 MIT15    EQU   *
BF69 30   C9 0082       LEAX  POINTER-BASE,U
BF6D 4D         MIT20   TSTA            WHILE NOT THIS POINTER DO
BF6E 27   05            BEQ   MIT30
BF70 30   02            LEAX  2,X       POINT AT NEXT POINTER
BF72 4A                 DECA
BF73 20   F8            BRA   MIT20

          BF75 MIT30    EQU   *
BF75 AE   84            LDX   ,X
BF77 AF   C9 0096       STX   CURPOI-BASE,U CURPOI -> EXPAND TEXT
BF7B 6C   41            INC   INPARM-BASE,U EXPANDING A PARAMETER

                        * next char comes from parm buffer

          BF7D MITFPA   EQU   *
BF7D AE   C9 0096       LDX   CURPOI-BASE,U
BF81 A6   80            LDA   ,X+       NEXT CHAR FROM BUFFER
BF83 AF   C9 0096       STX   CURPOI-BASE,U UPDATE POINTER

BF87 35   10            PULS  X         RESTORE X
BF89 4D                 TSTA            IF NOT END OF PARM THEN
BF8A 27   01            BEQ   MITF10
BF8C 39                 RTS             READY

          BF8D MITF10   EQU   *         OTHERWISE
BF8D 6F   41            CLR   INPARM-BASE,U NOT READING A PARM
BF8F 20   B3            BRA   MITIN

BF91 3F 20 49 6C  MITILR FCC  '? Illegal reference to parameter'
BF95 6C 65 67 61
BF99 6C 20 72 65
BF9D 66 65 72 65
BFA1 6E 63 65 20
BFA5 74 6F 20 70
BFA9 61 72 61 6D
BFAD 65 74 65 72
BFB1 20 69 6E 20          FCC   ' in command file'
BFB5 63 6F 6D 6D
BFB9 61 6E 64 20
BFBD 66 69 6C 65
BFC1 04                   FCB   4

                        *
                        * Proc MIFMS reads the next char from
                        * the command file, processing literals
                        *
          BFC2 MIFMS    EQU   *
BFC2 6F   C4            CLR   LITERA-BASE,U PRESET: NOT LITERAL
BFC4 8D   00            BSR   MYFMS     GET CHAR
BFC6 81   40            CMPA  #'@       IF IT'S A LITERAL-MARKER
BFC8 26   04            BNE   MIFM1
BFCA 6C   C4            INC   LITERA-BASE,U THEN SET FLAG
BFCC 8D   05            BSR   MYFMS     AND GET LITERAL CHAR
          BFCE MIFM1    EQU   *
BFCE A7   C9 0098       STA   LSTCHR-BASE,U KEEP CHAR
BFD2 39                 RTS

                        *
                        * MY-FMS: private FMS call
                        *
          BFD3 MYFMS    EQU   *
BFD3 34   10            PSHS  X
BFD5 30   C9 00A2       LEAX  MYFCB-BASE,U
BFD9 3B   0406          JSR   FMS       CALL FMS
BFDC 26   02            BNE   MYFM10    IF NO ERROR THEN
```

```
BFDE 35  90            PULS  I,PC      DONE

          BFE0 MYFM10  EQU   *
BFE0 A6   01           LDA   FCBERR,I  PICKUP ERROR #BR
BFE2 81   08           CMPA  #FPGEOF   IF NO END OF FILE THEN
BFE4 27   03           BEQ   MYFM20
BFE6 16   FE23         LBRA  DKERR     REPORT ERROR AND EXIT

          BFE9 MYFM20  EQU   *
BFE9 16   FE42         LBRA  QUIT      END OF FILE: NO ERROR
                       END   START

0 ERROR(S) DETECTED

SYMBOL TABLE:

ABTERR BE6A  BASE   BC00  BINFIL BED0  BREAK  BE5A  BRKCHR 0001
BUFFER BC02  BUFLEN 0080  CAFTER BC9C  CBFPTR BC98  CKME10 C1B8
CKMEME C179  CODARE BC00  CR     000D  CURLNR CC1A  CURPOI BC96
DCNFLG CC28  DEFDRV CC0D  DEFEXT C103  DKER1  C13B  DKERR  BE0C
DOCMND CB4B  DOCOMM BF0B  DOIT10 C125  ECHO   BEF6  ECHO10 BEF8
ECHO20 BF04  FCBACT 0002  FCBERR 0001  FCBEXT 000C  FCBFCB 0080
FCBLEN 0140  FMCLOS 0004  FMOPER 0001  FMS    0406  FMSCLS B403
FPAEOF 0008  GETFIL CD2B  GOLOOP C172  ILLFIL BEE2  IN10   C1F8
INCH   CB09  INCH2  CB0C  INITCB C1E7  INPARM BC01  KB10   BE54
KBBIN  BE4C  LBFPTR CC14  LF     000A  LITERA 8000  LOPART BDE2
LSTCHR BC9A  MEMEMO CC2B  MEMERR C190  MYFCB  BCA2  MYFM10 BFE0
MYFM20 BFE9  MYFMS  BFB3  NAPARM 000A  NIERRA C254  NIFMI  BFCE
NIFMS  BFC2  NIPA10 C24C  NIPARM C234  NIT05  BF54  NITIO  BF5A
NIT15  BF69  NIT20  BF6B  NIT30  BF75  NITC10 BECC  NITCH  C027
NITCHR BE5A  NITCNL C1E6  NITCNL C184  NITFIO BFB0  NITFPA BF7D
NITILR BF91  NITIN  BF44  NITL10 BBEC  NITLIN BDE3  PARAM1 BF2F
PARAM2 BF38  PARAM3 BF42  PARAMI BF15  PCRLF  C024  PLUSFL BC9B
POINTE BCB2  PRV10  BE22  PRVERR BE1A  PSCHB  BC9D  PSTRNG CD1E
PTCP10 C21B  PTCPAR C20A  PTCPER C21E  PUTCHR CD1B  QUIT   BE2E
RBCH10 C1AF  RBCH20 C1B7  RBCH30 C1C1  RBCH40 C1CB  RBCHOL C1AC
RDCMEI C100  RDL10  BE9C  RDL20  BEAC  RDLINE BE7E  RPTERR CD3F
SAVSTK BC9E  SPECOM C13F  START  C106  SWNEXT BCA0  TRKSTA CB4E
TTPAUS CC09  VN     0003  WARMS  C003  WRBUS  BF62  ITCOMP C14F
```

ELEKTRA SUPER FLOPPY CONTROLLER

'68' Micro Journal
5900 Cassandra Smith
P.O. Box 849
Hixson, TN 37343

Dear Don,

The enclosed disk contains a Product Review of the new AAA Chicago Computer Center ELEKTRA Super Floppy Controller disk controller board. Suffice to say, the product WORKS AS ADVERTISED.

Thanks for a terrific magazine in '68' Micro Journal. I have been a subscriber since August 1980 when I first became interested in microcomputers. It was your magazine that prompted me to inquire to SWTPc when I was ready to move up from my Heathkit ET-3400. I have since purchased numerous products from SS-50 vendors based on their advertisements in '68' Micro Journal.

I am a Communications-Electronics technician by trade, and really appreciated your notes in the November 1984 issue concerning the book review and hobbyists. I used to subscribe to Kilobaud/Microcomputing back in the days when Peter Stark wrote his "Thoughts on the SWTP Computer" column, but that megezine rapidly forgot about the SWTP and the SS-50 world in general. I also "forgot" to renew my subscription.

KEEP UP THE GOOD WORK at '68' MICRO JOURNAL.

Sincerely,

s/Michael R. Turner

4900 E. 5th St. 202
Tucson, AZ 85711
(602) 790-6355

- - -

PRODUCT REVIEW - ELEKTRA SUPER FLOPPY CONTROLLER

Michael R. Turner

The Elektra (tm) Super Floppy Controller is an SS-50 disk controller based on the Western Digital 2793 controller chip. To quote the manual "You have just purchased the best PIO disk controller available for the SS-50 bus." I cannot agree more.

I ordered the board from AAA Chicago Computer Center, 120 Chestnut Lane, Wheeling, IL. 60090, in September 1983. Since the board was not ready for shipment, I purchased a SWTPc DC-4 board to tide me over. Although I eventually kept both boards, AAA offered a full refund on the DC-4 board when the SFC was shipped in January 1984.

THE HARDWARE

The PCB is a professional, double-sided, plated thru board that is fully silkscreened. I should add that I purchased the board fully assembled and tested. AAA also offers the SFC as a bare board. As stated, the controller chip is the Western Digital 2793, which contains an on-chip phase locked loop data separator. The board contains 13 option jumpers (more on those later) that are easily identified and accessible for programming. Gold Molex connectors are used for the bus connectors. Two cable connectors, 34 pin for 5 inch and 50 pin for 8 inch drives, are provided. One nice feature of the design is that there is a test connector for connecting test equipment to the board for use during alignment or field service.

THE DOCUMENTATION

The documentation received consists of an 8 page Installation Manual, a 34 page Technical Manual, and 7 pages of TSC Utility style explanation of the FORMAT and SETUP utilities.

The Installation Manual provides a general description of the board, the software drivers, several "don't do" type cautions, and instructions for creating a bootable version of FLEX (tm) using FLEX.COR and the supplied drivers. If you are using 8 inch drives, the installation manual will get you going. For use on 5 inch drives, you will have to read on into the Technical Manual. I recommend this approach regardless of the size or type of drives used.

The Technical Manual is well written and thorough. It provides a hardware overview, board configuration and adjustments, software programming, a parts list, component overlay, and a schematic. In short, all of the documentation we have become accustomed to from SS-50 suppliers. I must say however, that AAA has gone one step further in providing detailed technical discussions concerning the jumper options, standard configurations, adjustments, unique SFC registers, WD 2793 registers (to include required WD delay times), and examples

of typical read and write sector routines.

OPTION PROGRAMMING

The SFC has 13 option jumpers that must be configured before installing the board in the computer. As mentioned, each option is thoroughly explained, including any interaction with other options. Also included are what AAA calls "3 Standard Configurations": Standard Elektra, GIMIX 28, and SWTPc DC-2. The full performance of the board is only available in the Standard Elektra Configuration. The GIMIX and SWTPC setups allow the user to prepare a bootable version of FLEX or OS9 (tm) for use in the Elektra configuration, as well emulation of the aforementioned boards. There is a caution statement that advises that improper jumper selections could damage the SFC circuitry or drives. Due to the detailed discussion and instructions, I doubt that any problems would arise unless a user has a system that uses totally non-standard drives. In any case, AAA is only a telephone call away and are always ready to help.

OPERATION

My system consists of a "stock block" SWTPc 69/K running at 1 MHZ, HUMBUG-09 (tm), and 6809 General FLEX. (As Jerry at AAA put it, I have about as standard a SWTPc as could be found). My drives are Tandon TM 100-2A's housed in two Elektra HD-5 dual drive cabinets, with power supplies (another excellent Elektra product).

After preparing a bootable system disk and setting up the Standard Elektra Configuration, I was set. The first thing to do is to prepare a STARTUP file that contains the SETUP options that the user wants. The following options (not all inclusive) are available:

- Drive size (5,8, Super 5, RAM disk, Winchester)
- Drive
- Number of sides
- Track stepping rate
- Single/Double track stepping (48 or 96 TPI)

The SETUP utility builds a drive parameter table for each drive in the system that is utilized by the SFC disk drivers. This is an interesting feature that allows a mix of 5 or 8 inch, single or double sided, 48 TPI or 96 TPI 5 inch drives to be used in the same system. When the system is first booted, the track access time defaults to 30 ms to read in the FLEX loader. When the STARTUP file is read, the SETUP parameters are then installed.

The user can examine or alter any of the SETUP parameters at any time simply by typing on the command line: SETUP <drive number>. One reason that I have found this feature to be valuable is that I run my drives at 6 ms track stepping, but encounter occasional read errors when reading distribution disks from vendors. By simply adjusting the track stepping time, I have no problems reading these disks. Once I have a working copy of the vendors disk, I reset the track stepping to 6 ms.

The FORMAT utility is used to create new disks. It uses information in the system parameter tables created by SETUP to set default parameters for the new disk. These parameters can be viewed or altered simply by typing on the command line: FORMAT <drive number>, or specified right on the command line: FORMAT,1,SS,SD. The FORMAT.CMD is not compatible with the TSC or SWTPc NEWDISK.CMD for the reason discussed above. A nice feature of the FORMAT.CMD is that when the formatting is complete, the FORMAT parameters are displayed on the terminal and the user is asked if they are correct. Typing an 'A' or 'Q' exits the formatter. Typing a 'Y' or 'N' allows multiple disks of the same format to be initialized without having to specify the parameters over and over again, or building an EXEC file in FLEX.

COMPATIBILITY

Since installation in January 1984, I have encountered no problems with compatibility with either my own older disks or new disks from vendors, other than the track stepping time mentioned above. On the software side, again no compatibility problems. I have several pieces of commercial software as well as that which I have wrote myself when using the SWTPc DC-4, and all of it runs unmodified. Also, I have used numerous programs and utilities from '68' Micro Journal with absolutely no problems. AAA has adhered to the TSC specifications well.

Environmentally, I am especially pleased with the SFC. Using the DC-4, I would experience sudden and repetitive disk read errors after about 4 or 5 hours of continuous operation. The remedy was to remove the cover from the computer and let the DC-4 air out (I think I've read about that some time ago in '68' Micro Journal). Actually, I don't believe it's the DC-4's fault. That board was designed for S/09+ systems which have much better air circulation than the 69/K. Since my workroom is not a "computer" room, the temperature can vary from 55 to 90 degrees. To date, I haven't experienced any temperature related problems. Humidity, on the other hand, is a problem all to itself.

CONCLUSION

I would heartily recommend the Elektra Super Floppy Controller to anyone, hobbyist or industrial user. For the inexperienced user, the initial installation and setup may take two hours; for experienced users, around 45 minutes (depending on how fast you read). The documentation is excellent, and for a die-hard hardware hacker like myself, thoroughly detailed. Since my career often relocates me to isolated areas and foreign countries, this type of detailed documentation is a necessity. I congratulate Jerry Koppel at the AAA Chicago Computer Center and Elektra for another outstanding product THAT WORKS AS ADVERTISED. Thanks Jerry.

MICHAEL R. TURNER

ELEKTRA is a trademark of AAA Chicago Computer Center.
FLEX is a trademark of Technical Systems Consultants.
OS9 is a trademark of Motorola, Inc. and Microware Sys
HUMBUG-09 is a trademark of Star Kits Software Syste

PROMPTED FILE TRANSFER

PROMPTED FILE TRANSFER

J. Gary Mills
1019 Weatherdon Ave.
Winnipeg, Manitoba
Canada R3M 2B5

Have you ever tried to transfer a file between computers, using modems, or between a micro computer and a large mainframe computer? Simple transfer methods, using existing hardware and software, sometimes work and sometimes don't. The reason is that the connection, where one system acts as a host

and the other simulates a remote terminal, is designed for hand typing and uses printed prompts to tell the user when to type.

The Problem

I have a smart terminal program which I use to communicate with a large Amdahl computer. It is quite easy to download a file by using a LIST command on the host and capturing the data in the memory of my micro. Later, I can save the data from memory to disk. Of course, the size of the file I can transfer is limited by the memory installed in my micro. If I attempt to upload a file from my micro to the host, using the INPUT command of it's editor, only part of the data gets captured. The same thing happens when someone calls my computer and tries to upload a file to BUILD. The causes of all these problems are pauses for disk activity, pauses for processing, or the prompts that editors provide for interactive data entry. This means that you can't just transmit a file, line after line, in a continuous stream of data. The editor can only accept a line of the file when it is ready for that line.

The Solution

One solution to all this confusion is prompted exchange of lines of data. It requires one computer to be a file sender and the other, a receiver. The most convenient arrangement is for the terminal emulator to act as sender and the host computer to act as receiver. For each line, the receiver signals that it is ready by transmitting a special control character called the prompt character. The sender then transmits a line of ASCII text followed by a RETURN, just like you would do from the keyboard. The sender can pause for disk activity before or during transmission of the line. When the receiver detects the RETURN, it processes the line and can pause for disk activity. The receiver can also transmit linefeeds, returns, or an ASCII message at this point. Then, when it is ready for another line, it transmits the prompt character. This is actually a simple "line turnaround" protocol using two control characters. The two computers take turns transmitting data and exchange roles by sending the control characters. The prompt character should be a character that doesn't occur within the data being exchanged.

Once all the lines of the file have been transmitted, it is necessary to indicate that the end of file has been reached, so that the receiver can close the file. The sender could transmit a special control character, such as EOT, to mark the end of file. Another alternative would be a special string, such as "END" followed by RETURN. The editors I use on the Amdahl computer require a null line, that is, just a RETURN, to terminate input to a file. The end of file marker should also be something that doesn't occur within the data being exchanged. In particular, the sender should pad null lines with one blank, when a null line is used to indicate end of file.

When doing a file transfer between computers, it is important to be sure that the file was received without errors such as those caused by noise on a phone line. Why not make use of the parity bit that accompanies the ASCII characters? Most ACIA driver routines do not check for parity errors, so a little software modification is needed here (see Listing 1). When a file receiver detects a parity error, one of the following actions could be taken: (1) Accept the bad character. (2) Mark the bad character with a "?" character. (3) Delete the line containing the bad character and instruct the sender to re-transmit the line. This ensures that only good data are stored in the file. This option requires another special control character which I call the retry character. For example, the BEL character could be transmitted just before the prompt character to request that the last line be re-transmitted.

Prompted file transfer is a great improvement in file transfer methods, particularly if parity checking is used, but it is not completely foolproof. If one of the control characters gets garbled in transmission, the transfer will hang up and require user intervention. There are solutions to this, such as acknowledgement of each line and timeouts in wait loops, but the condition does not arise often enough to be a problem. Of course, a system designed for automatic unattended operation would need to be completely bulletproof.

The Programs

Based on the specifications outlined above, the first program I wrote was RECEV.CMD (see Listing 2). It runs on my host system, which is a modification of Flex to support an auto-answer modem. It works like the BUILD command, prompting with an equal sign before the control character prompt for each line. All of the file transfer parameters can be changed to suit the wishes of the user.

My friend, Scott Fraser, helped in the final testing of RECEV. He called my computer, tried a few normal Flex commands, and then entered the RECEV command, specifying a new file name. He set the transfer parameters to match the ones he had previously set up at his end. Then, RECEV prompted for the first line, and transfer began. Everything worked perfectly. At times, the transfer paused briefly, as RECEV wrote sectors to the disk. At end of file, the last sector was written, and Scott again saw the Flex prompt.

I wrote SEND.CMD as a demonstration of the technique of prompted file transfer (see Listing 3). It works like the LIST command, except that it needs a control character prompt before it will display each line of the file. All the transfer parameters can be changed just like in the receiver. To be useful, it needs to be integrated into a terminal communication program. Brief instructions on how to do this are included as comments.

I can now upload files from Flex disks to the Amdahl mainframe computer. Once I have signed on to the time sharing system, I enter a command that tells it to prefix each record read from the terminal with a DC1 control character. Then, I type the INPUT command of the editor and simultaneously set the transmit flag of my terminal program. When I press RETURN, the transfer begins. It is amazing to watch the interactive conversation between the two machines. At end of file, the transfer stops, and I am back in control again.

Listing 1: Parity-checking input routine.

```
* SEND COMMAND
*   A PROMPTING FILE SENDER DEMONSTRATION
*   FLEX9 VERSION
*
*************************************
*                                   *
* AUTHOR:     J. GARY MILLS         *
*             1019 HEATHERDOWN AVE. *
*             WINNIPEG, MANITOBA    *
*             CANADA    R3R 2B5     *
```

```
                        *                                               *
                        *************************************************
                        *
                        *
                        * ADAPTATION TO A MODEM COMMUNICATION PROGRAM:
                        *   -DELETE UPPER LEVEL ROUTINE 'SESTAR'
                        *   -CALL 'FSETUP' AND 'PSETUP' FROM PROGRAM
                        *   -EXAMINE 'FTRANS' FOR ENTRY REQUIREMENTS OF 'SLINE'
                        *   -DELETE 'FTRANS'
                        *   -CALL 'SLINE' FROM PROGRAM
                        *   -REDIRECT OUTPUT OF 'SLINE' TO MODEM PORT
                        *   -PROVIDE CONTROL FOR THE FLAG 'FTRAN'
                        *
                        *
                        *
                        * FLEX ADDRESSES
C100  UCS     EQU   $C100     UTILITY COMMAND SPACE
C840  FCB     EQU   $C840     FLEX FCB
CC00  TTYBS   EQU   $CC00     TTYSET BACKSPACE CHAR
CC01  TTYDEL  EQU   $CC01     TTYSET DELETE CHAR
CC14  LINEPT  EQU   $CC14     LINE BUFFER POINTER
CD2D  GETFIL  EQU   $CD2D     PARSE FILE SPEC
CD33  SETEXT  EQU   $CD33     SET EXTENSION
CD03  WARMS   EQU   $CD03     FLEX RETURN POINT
CD3F  RPTERR  EQU   $CD3F     REPORT ERROR
CD1E  PSTRNG  EQU   $CD1E     PRINT NL AND STRING
CD15  GETCHR  EQU   $CD15     GET A CHAR
CD18  PUTCHR  EQU   $CD18     PUT A CHAR
CD21  CLASS   EQU   $CD21     CLASSIFY CHARACTER
CD24  PCRLF   EQU   $CD24     PRINT NEWLINE
CD3C  OUTHEX  EQU   $CD3C     PRINT HEX NUMBER
CD39  OUTDEC  EQU   $CD39     PRINT DECIMAL NUMBER
0406  FMS     EQU   $0406     FLEX DOS
                        *
                        *
                        *
                        * PROGRAM CONSTANTS
00FF  LIMIT   EQU   255       LINE BUFFER LENGTH
0001  ACCEPT  EQU   1         ERROR OPTION CODES
0002  WARN    EQU   2
0003  RETRY   EQU   3
                        *
                                *
                                *
C100                            ORG   UCS
C100 20  0B             SEND    BRA   SESTAR
C102 01                         FCB   1         VERSION NUMBER
                        * VARIABLES
C103 00                 FTRAN   FCB   0         TRANSMIT FLAG
C104 11                 PROCH   FCB   $11       PROMPT CHAR
C105 07                 RTRCH   FCB   $07       RETRY CHAR
C106 04                 EOFCH   FCB   $04       END OF FILE CHAR
C107 01                 PEOPT   FCB   1         PARITY ERROR OPTION
C108 0D 00 00 00        EOFST   FCB   $0D,0,0,0,0 END OF FILE STRING
C10C 00
                                *
                                *
                                * SEND A FILE
C10D BD  15             SESTAR  BSR   FSETUP    PARSE FILESPEC AND OPEN
C10F 25  10                     BCS   SEEND     IF NO ERROR
C111 7C  C103                   INC   FTRAN     . SET TRANSMIT FLAG
C114 BD  34                     BSR   PSETUP    . OBTAIN PARAMETERS
C116 8E  C3A9                   LDX   #MESREA
C119 BD  5E                     BSR   STRING    . PRINT READY MESSAGE
C11B BD  C20A                   JSR   PRNL      . START NEW LINE
C11E BD  C319                   JSR   FTRANS    . TRANSFER FILE
C121 7E  C003           SEEND   JMP   WARMS     GOTO FLEX
                                *
                                *
                                * PARSE FILESPEC AND OPEN FILE
C124 8E  CC14           FSETUP  LDX   LINEPT    EXAMINE LINE BUFFER
C127 A6  84                     LDA   0,X
C129 BD  CD21                   JSR   CLASS
C12C 25  19                     BCS   FSSEC     IF OPERAND EXISTS
C12E BD  31                     BSR   PARFIL    . PARSE FILE NAME
C130 25  15                     BCS   FSSEC     . IF NO ERROR
C132 BD  C20A                   JSR   PRNL      .   START NEW LINE
C135 8E  C840                   LDX   #FCB      .   POINT TO FCB
C138 B6  01                     LDA   #1
C13A A7  84                     STA   0,X
C13C BD  C3A6                   JSR   FILRS     .   OPEN FOR READ
C13F 26  03                     BNE   FSEROP    .     IF NO ERROR
C141 1C  FE             FSCLC   ANDCC #$FE      .       CLEAR FLAG
C143 39                         RTS             .     ELSE
C144 BD  C3A3           FSEROP  JSR   ERMES     .       REPORT ERROR
C147 1A  01             FSSEC   ORCC  #$01      .       SET FLAG
C149 39                         RTS             RETURN
                                *
                                *
                                * DISPLAY AND CHANGE TRANSFER PARMS
C14A BD  C20A           PSETUP  JSR   PRNL
C14D BD  20             PSPRM   BSR   DISPRM    REPEAT; DISPLAY PARMS
C14F 8E  C387                   LDX   #MESCHA   . PROMPT FOR CHANGES
C152 BD  25                     BSR   STRING
C154 BD  CIF0                   JSR   UPCHAR    . GET REPLY
C157 81  4E                     CMPA  #'N       UNTIL "NO"
C159 27  05                     BEQ   PSRET
C15B BD  C20D                   JSR   CHAPRM    . CHANGE PARMS
C15E 20  ED                     BRA   PSPRM     LOOP
C160 39                 PSRET   RTS             RETURN
                                *
                                * PARSE FILE NAME
C161 8E  C840           PARFIL  LDX   #FCB      POINT TO FCB
C164 BD  CD2D                   JSR   GETFIL    PICK UP FILE NAME
C167 25  08                     BCS   PAERFI    IF NO ERROR
C169 86  01                     LDA   #1
C16B BD  CD33                   JSR   SETEXT    . DEFAULT := .TXT
C16E 1C  FE                     ANDCC #$FE      . CLEAR FLAG
C170 39                         RTS
C171 8E  C3CC           PAERFI  LDX   #MESFIL   ELSE
C174 BD  03                     BSR   STRING    . REPORT ERROR
C176 1A  01                     ORCC  #$01      SET FLAG
C178 39                         RTS             RETURN
                                *
                                *
                                * BEGIN NEW LINE AND PRINT STRING
C179 7E  CD1E           STRING  JMP   PSTRNG
                                *
                                *
                                * DISPLAY TRANSFER PARAMETERS
C17C BD  C20A           DISPRM  JSR   PRNL      BEGIN NEW LINE
C17F 8E  C3DE                   LDX   #MESTRA   DISPLAY TITLE
C182 BD  F5                     BSR   STRING
C184 8E  C3F3                   LDX   #MESPRO   DISPLAY PROMPT CHARACTER
C187 BD  59                     BSR   INDENT
C189 8E  C104                   LDX   #PROCH
C18C BD  5F                     BSR   BYTE
C18E 8E  C403                   LDX   #MESRET   DISPLAY RETRY CHAR
C191 BD  4F                     BSR   INDENT
C193 8E  C105                   LDX   #RTRCH
C196 BD  55                     BSR   BYTE
C198 7D  C106                   TST   EOFCH     IF CONTROL CHAR
C19B 2B  0C                     BMI   DISTR
C19D 8E  C412                   LDX   #MESEOF   . DISPLAY END OF FILE CHAR
C1A0 BD  40                     BSR   INDENT
C1A2 8E  C106                   LDX   #EOFCH
C1A5 BD  46                     BSR   BYTE
C1A7 20  1A                     BRA   DIPEO     ELSE
C1A9 8E  C427           DISTR   LDX   #MESSTR   . DISPLAY END OF FILE STRING
C1AC BD  34                     BSR   INDENT
C1AE 86  22                     LDA   #'"       . OPEN QUOTE
C1B0 BD  55                     BSR   SCHAR
C1B2 8E  C108                   LDX   #EOFST
C1B5 A6  80             DISPST  LDA   ,X+
C1B7 81  00                     CMPA  #$00
C1B9 27  04                     BEQ   DIQUO
C1BB BD  0A                     BSR   SCHAR
C1BD 20  F6                     BRA   DISPST
C1BF 86  22             DIQUO   LDA   #'"       . CLOSE QUOTE
C1C1 BD  44                     BSR   SCHAR
C1C3 8E  C43D           DIPEO   LDX   #MESPEO   DISPLAY ERROR OPTION
C1C6 BD  1A                     BSR   INDENT
C1C8 8E  C454                   LDX   #MESACC
C1CB B6  C107                   LDA   PEOPT
C1CE 81  03                     CMPA  #RETRY
C1D0 26  05                     BNE   PIWRN
C1D2 8E  C460                   LDX   #MESRTR
C1D5 20  07                     BRA   PIEOC
C1D7 81  02             PIWRN   CMPA  #WARN
C1D9 26  03                     BNE   PIEOC
C1DB 8E  C458                   LDX   #MESWAR
```

```
C1DE 8D  10      PIEOC   BSR    SEGMNT
C1E0 20  20              BRA    PRNL      BEGIN NEW LINE; RETURN

                 *
                 *
                 * PRINT S(I) SPACES & I-->STRING
C1E2 34  10      INDENT  PSHS   X
C1E4 8E  C466            LDX    #MESSPA
C1E7 8D  90              BSR    STRING
C1E9 35  10              PULS   X
C1EB 20  03              BRA    SEGMNT
                 *
                 *
                 * PRINT X-->BYTE IN HEX
C1ED 7E  CD3C    BYTE    JMP    OUTHEX
                 *
                 *
                 * PRINT X-->STRING
C1F0 A6  80      SEGMNT  LDA    ,X+
C1F2 81  04              CMPA   #$04
C1F4 27  04              BEQ    SERET
C1F6 8D  0F              BSR    SCHAR
C1F8 20  F6              BRA    SEGMNT
C1FA 39          SERET   RTS
                 *
                 *
                 * INPUT A CHARACTER IN UPPER CASE
C1FB 8D  07      UPCHAR  BSR    ACHAR
C1FD 81  60              CMPA   #$60
C1FF 23  02              BLS    *+4
C201 84  0F              ANDA   #$0F
C203 39                  RTS
                 *
                 *
                 * INPUT A CHARACTER
C204 7E  CD15    ACHAR   JMP    GETCHR
                 *
                 *
                 * PRINT A CHARACTER
C207 7E  CD18    SCHAR   JMP    PUTCHR
                 *
                 *
                 * BEGIN NEW LINE
C20A 7E  CD24    PRNL    JMP    PCRLF
                 *
                 *
                 * CHANGE TRANSFER PARAMETERS
C20D 8D  FB      CHAPRM  BSR    PRNL      BEGIN NEW LINE
C20F 8E  C46D            LDX    #MESPRE   DISPLAY INSTRUCTION
C212 8D  6D              BSR    STRING
C214 8D  F4              BSR    PRNL      BEGIN NEW LINE
C216 8E  C3F3            LDX    #MESPRO   PROMPT FOR PROMPT CHAR
C219 8D  67              BSR    CONTR     GET REPLY
C21B 25  03              BCS    CHRTN     IF NOT NULL
C21D B7  C104            STA    PROCH     . INSERT PROMPT CHAR
C220 8E  C403    CHRTN   LDX    #MESRET   PROMPT FOR RETRY CHAR
C223 8D  5B              BSR    CONTR     GET REPLY
C225 25  03              BCS    CHEFT     IF NOT NULL
C227 B7  C105            STA    RTYCH     . INSERT RETRY CHAR
C22A 8E  C482    CHEFT   LDX    #MESEFT   REPEAT; PROMPT FOR EOF TYPE
C22D 8D  50              BSR    STRING

C22F 8D  CA              BSR    UPCHAR    . GET REPLY
C231 81  0D              CMPA   #$0D
C233 27  2A              BEQ    CHPEO     UNTIL CR
C235 81  43              CMPA   #'C
C237 26  0C              BNE    CHTST     . IF 'C'
C239 8E  C412            LDX    #MESEOF   .   PROMPT FOR EOF CHAR
C23C 8D  44              BSR    CONTR     .   GET REPLY
C23E 25  1F              BCS    CHPEO     UNTIL NULL
C240 B7  C106            STA    EOFCH     .   INSERT EOF CHAR
C243 20  1A              BRA    CHPEO     . ELSE
C245 81  53      CHTST   CMPA   #'S
C247 26  08              BNE    CHTBU     . IF 'S'
C249 8E  C427            LDX    #MESSTR   .   PROMPT FOR EOF STRING
C24C 8D  C2E4            JSR    WORD      .   GET REPLY
C24F 20  0E              BRA    CHPEO     . ELSE
C251 81  4E      CHTBU   CMPA   #'N
C253 26  05              BNE    CHEFT     . IF 'N'
C255 86  09              LDA    #$09
C257 B7  C108            STA    EOFST     .   INSERT NULL STRING
C25A 86  FF              LDA    #$FF      .   SET TYPE FLAG
C25C B7  C106            STA    EOFCH     UNTIL VALID REPLY
C25F 8E  C405    CHPEO   LDX    #MESOPT   REPEAT
C262 8D  1B              BSR    STRINV    . PROMPT FOR ERROR OPTION
C264 8D  95              BSR    UPCHAR    . GET REPLY
C266 81  0D              CMPA   #$0D
C268 27  13              BEQ    CHRET     UNTIL NULL
C26A C6  01              LDB    #1
C26C 81  41              CMPA   #'A       . IF 'A'
C26E 27  0A              BEQ    CHSEO     .   OPT := ACCEPT
C270 5C                  INCB
C271 81  4D              CMPA   #'M       . ELIF 'M'
C273 27  05              BEQ    CHSEO     .   OPT := MARK
C275 5C                  INCB
C276 81  52              CMPA   #'R       . ELIF 'R'
C278 26  E5              BNE    CHPEO     .   OPT := RETRY
C27A F7  C107    CHSEO   STB    PEOPT     UNTIL VALID REPLY
C27D 20  8D      CHRET   BRA    PRNL      BEGIN NEW LINE; RETURN
C27F 7E  C179    STRINV  JMP    STRING
                 *
                 *
                 * PROMPT AND HEX BYTE INPUT
C282 1F  12      CONTR   TFR    X,Y
C284 1F  21      COPRO   TFR    Y,X       REPEAT
C286 8D  F7              BSR    STRINV    . PROMPT FOR INPUT
C288 5F                  CLRB             . COUNT := 0
C289 34  04              PSHS   X         . RESULT := 0
C28B 34  04              PSHS   B         . ERROR FLAG := 0
C28D 8D  CIFB    COCHA   JSR    UPCHAR    . REPEAT; GET A CHAR
C290 81  08              CMPA   #$08
C292 27  30              BEQ    COLEX     . UNTIL CHAR = CR
C294 81  CC01            CMPA   ??YDEL
C297 27  EB              BEQ    COPRO     . UNTIL CHAR = DELETE
C299 81  CC00            CMPA   ??YBS
C29C 26  0F              BNE    COCOU     .   IF CHAR = BS
C29E 5A                  DECB             .     DECR COUNT
C29F 2B  E3              BMI    COPRO     . UNTIL COUNT < 0
C2A1 64  E4              LSR    0,S
C2A3 A6  61              LDA    1,S       .       SHIFT OUT 1 DIGIT
C2A5 44                  LSRA
C2A6 44                  LSRA
C2A7 44                  LSRA
C2A8 44                  LSRA
C2A9 A7  61              STA    1,S
C2AB 20  E0              BRA    COCHA     .   ELSE
C2AD 5C          COCOU   INCB             .     INCR COUNT
C2AE C1  03              CMPB   #3        .     IF COUNT < 3
C2B0 2C  08              BGE    COCHA
C2B2 8D  1B              BSR    HEXDIG    .       CONVERT CHAR
C2B4 69  E4              ROL    0,S
C2B6 68  61              ASL    1,S       .       SHIFT DIGIT TO RESULT
C2B8 68  61              ASL    1,S
C2BA 68  61              ASL    1,S
C2BC 68  61              ASL    1,S
C2BE AB  61              ADDA   1,S
C2C0 A7  61              STA    1,S
C2C2 20  C9              BRA    COCHA     . LOOP
C2C4 6D  E4      COLEX   TST    0,S
C2C6 26  BC              BNE    COPRO     UNTIL VALID
C2C8 A6  61              LDA    1,S       GET RESULT
C2CA 32  62              LEAS   2,S
C2CC C0  01              SUBB   #1        IF NULL: SET CARRY
C2CE 39                  RTS              RETURN
                 *
                 *
                 * CONVERT CHAR TO HEX DIGIT
C2CF 80  30      HEXDIG  SUBA   #$30
C2D1 2B  0A              BMI    HEXOR
C2D3 81  09              CMPA   #$09
C2D5 23  02              BLS    HEXHE
C2D7 80  07              SUBA   #$07
C2D9 81  0F      HEXHE   CMPA   #$13
C2DB 23  04              BLS    HEXAL
C2DD 4F          HEXOR   CLRA             IF NOT HEX
C2DE 1A  01              ORCC   #$01       . SET CARRY
C2E0 39                  RTS
C2E1 1C  FE      HEXAL   ANDCC  #$FE
C2E3 39                  RTS              RETURN
                 *
                 *
```

```
                    * PROMPT AND STRING INPUT
C2E4 1F  12    WORB   TFR   S,Y
C2E6 1F  21    WOPRO  TFR   Y,X       REPEAT
C2E8 BD  95            BSR   STRINV   . PROMPT FOR INPUT
C2EA 5F                CLRB           . COUNT := 0
C2EB BE  C108          LDX   #EOFST   . POINT TO RESULT
C2EE 8D  C204  WOCHA   JSR   ACHAR    . REPEAT; GET A CHAR
C2F1 81  0D            CMPA  #$0D
C2F3 27  1A            BEQ   WOLEX    . UNTIL CHAR = CR
C2F5 81  CC01          CMPA  TTYDEL
C2F8 27  EC            BEQ   WOPRO    . UNTIL CHAR = DELETE
C2FA 81  CC00          CMPA  TTYBS
C2FD 26  07            BNE   WOCOU    .    IF CHAR = BS
C2FF 5A                DECB           .       DECR COUNT
C300 2B  E4            BMI   WOPRO    . UNTIL COUNT < 0
C302 30  1F            LEAX  -1,X
C304 20  E8            BRA   WOCHA    .    ELSE
C306 5C        WOCOU   INCB           .       INCR COUNT
C307 C1  05            CMPB  #5       .       IF COUNT < 5
C309 2C  E3            BGE   WOCHA
C30B A7  80            STA   ,X+      .          INSERT CHAR
C30D 20  DF            BRA   WOCHA    . LOOP
C30F B4  0D    WOLEX   LDA   #$0D     UNTIL VALID INPUT
C311 A7  84            STA   0,X      MARK END OF STRING
C313 86  FF            LDA   #$FF     SET TYPE FLAG
C315 B7  C106          STA   EOFCH
C318 39                RTS            RETURN
                    *
                    *
                    * PROMPTED FILE TRANSFER
C319 86  C105  FTRANS  LDA   RTRCH
C31C 4C                INCA
C31D 34  02            PSHS  A        INIT PREVIOUS CHAR
C31F 7D  C103  FTRPT   TST   FTRAN    REPEAT
C322 27  17            BEQ   FTRET    UNTIL FLAG CLEAR
C324 8D  C204          JSR   ACHAR    . GET A CHAR
C327 B1  C104          CMPA  PROCH    . IF CHAR = PROMPT CHAR
C32A 26  0B            BNE   FTOLD
C32C E6  E4            LDB   0,S
C32E F1  C105          CMPB  RTRCH    .    COMPARE RETRY CHAR
C331 34  02            PSHS  A
C333 8D  09            BSR   SLINE    .    SEND A LINE
C335 35  02            PULS  A
C337 A7  E4    FTOLD   STA   0,S      . SAVE CHAR
C339 20  E4            BRA   FTRPT    LOOP
C33B 32  61    FTRET   LEAS  1,S
C33D 39                RTS            RETURN
                    *
                    *
                    * SEND A LINE FROM FILE
C33E 26  07    SLINE   BNE   SLCOP    IF NOT RETRY
C340 B6  C107          LDA   PEOPT
C343 81  03            CMPA  #RETRY
C345 27  15            BEQ   SLSEN
C347 10BE C4EA SLCOP   LDY   #BUFFER
C34B BE  C040  SLREAD  LDX   #FCB     . REPEAT
C34E 8D  56            BSR   FILMS    .    GET CHAR FROM FILE
C350 26  32            BNE   SLERR    . UNTIL DISK ERROR
C352 81  0A            CMPA  #$0A     .    DISCARD LINEFEED
C354 27  F5            BEQ   SLREAD
C356 A7  A0            STA   ,Y+      .    INSERT IN BUFFER
C358 81  0D            CMPA  #$0A     . UNTIL CHAR = CR
C35A 26  EF            BNE   SLREAD   . LOOP
C35C 8E  C4EA  SLSEN   LDX   #BUFFER  IF NO ERROR
C35F 5F                CLRB
C360 A6  80    SLRPT   LDA   ,X+      . REPEAT;  GET CHAR FROM BUFFER
C362 5D                TSTB           .    IF NULL LINE AND NULL EOF STRING
C363 26  15            BNE   SLCHA
C365 81  0D            CMPA  #$0D
C367 26  11            BNE   SLCHA
C369 7D  C106          TST   EOFCH
C36C 2A  0C            BPL   SLCHA
C36E B1  C108          CMPA  EOFST
C371 26  07            BNE   SLCHA
C373 86  20            LDA   #$20     .       SEND A SPACE
C375 BD  C207          JSR   SCHAR
C378 86  0D            LDA   #$0D
C37A BD  C207  SLCHA   JSR   SCHAR    .    SEND CHAR
C37D 5C                INCB

C37E B1  0D            CMPA  #$0D     . UNTIL CHAR = CR
C380 26  DE            BNE   SLRPT    . LOOP
C382 20  1E            BRA   SLRET    ELSE
C384 7F  C103  SLERR   CLR   FTRAN    . CLEAR TRANSMIT FLAG
C387 A6  01            LDA   1,X
C389 B1  08            CMPA  #8       . IF END OF FILE
C38B 26  13            BNE   SLDEM
C38D 86  04            LDA   #4
C38F A7  84            STA   0,X
C391 8D  13            BSR   FILMS    .    CLOSE FILE
C393 86  C106          LDA   EOFCH    .    IF CONTROL CHAR EOF
C396 2B  03            BMI   SLCON
C398 7E  C207          JMP   SCHAR    .       SEND IT
C39B 8E  C108  SLCON   LDX   #EOFST   .    ELSE
C39E 20  C0            BRA   SLRPT    .       SEND EOF STRING
C3A0 8D  01    SLDEM   BSR   ERMES    . ELSE;  REPORT ERROR
C3A2 39        SLRET   RTS            RETURN
                    *
                    *
                    * REPORT DISK ERROR
C3A3 7E  CD3F  ERMES   JMP   RPTERR
                    *
                    *
                    * DOS FILE MANAGEMENT
C3A6 7E  D406  FILMS   JMP   FMS
                    *
                    *
                    * MESSAGES
C3A9 52 45 41 44 MESREA FCC   'READY TO SEND'
C3AD 59 20 54 4F
C3B1 20 53 45 4E
C3B5 44
C3B6 04                FCB   $04
C3B7 4D 41 4B 45 MESCHA FCC   'MAKE CHANGES (Y/N)? '
C3BB 20 43 48 41
C3BF 4E 47 45 53
C3C3 20 28 59 2F
C3C7 4E 29 3F 20
C3CB 04                FCB   $04
C3CC 49 4E 56 41 MESFIL FCC   'INVALID FILE NAME'
C3D0 4C 49 44 20
C3D4 46 49 4C 45
C3D8 20 4E 41 4D
C3DC 45
C3DD 04                FCB   $04
C3DE 54 52 41 4E MESTRA FCC   'TRANSFER PARAMETERS:'
C3E2 53 46 45 52
C3E6 20 50 41 52
C3EA 41 4D 45 54
C3EE 45 52 53 3A
C3F2 04                FCB   $04
C3F3 50 52 4F 4D MESPRO FCC   'PROMPT CHAR = $'
C3F7 50 54 20 43
C3FB 48 41 52 20
C3FF 3D 20 24
C402 04                FCB   $04
C403 52 45 54 52 MESRET FCC   'RETRY CHAR = $'
C407 59 20 43 48
C40B 41 52 20 3D
C40F 20 24
C411 04                FCB   $04
C412 45 4E 44 20 MESEOF FCC   'END OF FILE CHAR = $'
C416 4F 46 20 46
C41A 49 4C 45 20
C41E 43 48 41 52
C422 20 3D 20 24
C426 04                FCB   $04
C427 45 4E 44 20 MESSTR FCC   'END OF FILE STRING = '
C42B 4F 46 20 46
C42F 49 4C 45 20
C433 53 54 52 49
C437 4E 47 20 3D
C43B 20
C43C 04                FCB   $04
C43D 50 41 52 49 MESPEO FCC   'PARITY ERROR OPTION = '
C441 54 59 20 45
C445 52 52 4F 52
C449 20 4F 50 54
C44D 49 4F 4E 20
```

```
C451 3B 20
C453 04                 FCB   $04
C454 41 43 43 45  MESACC FCC   'ACCEPT'
C458 50 54
C45A 04                 FCB   $04
C45B 4D 41 52 4B  MESMAR FCC   'MARK'
C45F 04                 FCB   $04
C460 52 45 54 52  MESRTR FCC   'RETRY'
C464 59
C465 04                 FCB   $04
C466 20 20 20 20  MESSPA FCC   '     '
C46A 20 20
C46C 04                 FCB   $04
C46D 50 52 45 53  MESPRE FCC   'PRESS RETURN TO SKIP'
C471 53 20 52 45
C475 54 55 52 4E
C479 20 54 4F 20
C47D 53 4B 49 50
C481 04                 FCB   $04
C482 4E 55 4C 4C  MESEFT FCC   'NULL (N), STRING (S) OR CHAR (C) '
C486 20 28 4E 29
C48A 2C 20 53 54
C48E 52 49 4E 47
C492 20 28 53 29
C496 20 4F 52 20
C49A 43 48 41 52
C49E 20 28 43 29
C4A2 20
C4A3 46 4F 52 20         FCC   'FOR END OF FILE? '
C4A7 45 4E 44 20
C4AB 4F 46 20 46
C4AF 49 4C 45 3F
C4B3 20
C4B4 04                 FCB   $04
C4B5 41 43 43 45  MESOPT FCC   'ACCEPT (A), MARK (M) OR RETRY (R) '
C4B9 50 54 20 28
C4BD 41 29 2C 20
C4C1 4D 41 52 4B
C4C5 20 28 4D 29
C4C9 20 4F 52 20
C4CD 52 45 54 52
C4D1 59 20 28 52
C4D5 29 20
C4D7 46 4F 52 20         FCC   'FOR PARITY ERROR? '
C4DB 50 41 52 49
C4DF 54 59 20 45
C4E3 52 52 4F 52
C4E7 3F 20
C4E9 04                 FCB   $04
                        *
                        *
                        * LINE BUFFER
C4EA                    BUFFER RMB   LINLT
          C5E9          BUFEND EQU   *
                        *
                        *
                        *
                               END   SEND
```

0 ERROR(S) DETECTED

SYMBOL TABLE:

```
ACCEPT 0001  ACHAR  C204  BUFEND C5E9  BUFFER C4EA  BYTE   C1ED
CHAPRM C208  CHEFT  C22A  CHPE0  C25F  CHRET  C27B  CHRTR  C220
CKSED  C27A  CNTMA  C251  CNTST  C245  CLASS  C021  COCHA  C200
COC0U  C2A0  COLEI  C2C4  CONTR  C202  COPRB  C284  DIPE0  C1C3
DIPST  C1B5  DIGUO  C10F  DISPRM C17C  DISTR  C1A9  EOFCH  C106
EOFBT  C108  ERMES  C3A3  FCB    C840  FILMS  C3A6  FRS    B406
FSCLC  C141  FSEROP C144  FSETUP C124  FSSEC  C147  FTOLD  C357
FTRAN  C103  FTRANS C319  FTRET  C33D  FTRPT  C31F  GETCHR C015
GETFIL CD2B  HECHE  C2D9  HEXBR  C2BB  HEVAL  C2E1  HEXBIG C2CF
INBENT C1E2  LIMIT  00FF  LINEPT CC14  MARK   0002  MESACC C454
MESCHA C387  MESEFT C482  MESEOF C412  MESFIL C3CC  MESMAR C458
MESOPT C4B5  MESPED C43B  MESPRE C46D  MESPRO C3F3  MESREA C3A9
MESRET C403  MESRTR C460  MESSPA C466  MESSTR C427  MESTRA C3DE
OUTDEC CD39  OUTHEI CD3C  PAERFI C171  PARFIL C161  PCRLF  CD24
PEOPT  C107  PIEOC  C1D6  PINMA  C1D7  PRBL   C20A  PROCH  C104
PSETUP C14A  PSPRA  C14B  PSRET  C160  PSTRNG CD1E  PUTCHR CD18
RETRY  0003  RPTERR CD3F  RTRCH  C105  SCHAR  C207  SEEBD  C121
SEGMNT C1F0  SEND   C100  SERET  C1FA  SESTAR C10B  SETEXT CD33
SLCHA  C37A  SLCON  C398  SLCOP  C347  SLDEN  C3A0  SLERR  C3B4
SLINE  C33E  SLREAD C340  SLRET  C342  SLRPT  C360  SLSEN  C33C
STRING C179  STRING C27F  TTYBS  CC00  TTYDEL CC01  UCS    C100
UPCHAR C1FB  WARMS  CD03  WDCHA  C2EE  WOCON  C304  WOLEI  C30F
WOPRG  C2E6  WORD   C2E4
```

Listing 2: Prompting file receiver.

```
                        *
                        * RECEV COMMAND
                        *    A PROMPTING FILE RECEIVER
                        *    FLEX9 VERSION
                        *
                        ********************************************
                        *                                          *
                        * AUTHOR:       J. GARY MILLS              *
                        *               1019 WEATHERDON AVE.       *
                        *               WINNIPEG, MANITOBA         *
                        *               CANADA    R3M 2B5          *
                        *                                          *
                        ********************************************
                        *
                        *
                        *
                        * FLEX ADDRESSES
          C100 UCS      EQU   $C100     UTILITY COMMAND SPACE
          C840 FCB      EQU   $C840     FLEX FCB
          CC00 TTYBS    EQU   $CC00     TTYSET BACKSPACE CHAR
          CC01 TTYDEL   EQU   $CC01     TTYSET DELETE CHAR
          CD2B GETFIL   EQU   $CD2B     PARSE FILE SPEC
          CD33 SETEXT   EQU   $CD33     SET EXTENSION
          CD03 WARMS    EQU   $CD03     FLEX RETURN POINT
          CD3F RPTERR   EQU   $CD3F     REPORT ERROR
          CD1E PSTRNG   EQU   $CD1E     PRINT CR, AND STRING
          CD15 GETCHR   EQU   $CD15     GET A CHAR
          CD18 PUTCHR   EQU   $CD18     PUT A CHAR
          CD24 PCRLF    EQU   $CD24     PRINT NEWLINE
          CD3C OUTHEX   EQU   $CD3C     PRINT HEX NUMBER
          CD39 OUTDEC   EQU   $CD39     PRINT DECIMAL NUMBER
          B406 FMS      EQU   $B406     FLEX DOS
                        *
                        *
                        *
                        * PROGRAM CONSTANTS
          00FF LIMIT    EQU   255       LINE BUFFER LENGTH
          0001 ACCEPT   EQU   1         ERROR OPTION CODES
          0002 MARK     EQU   2
          0003 RETRY    EQU   3
                        *
                        *
                        *
C100                    ORG   UCS
C100 20  0E     RECEV   BRA   RESTAR
C102 01                 FCB   1         VERSION NUMBER
                        * VARIABLES
C103 00         PFLAG   FCB   0         PARITY ERROR FLAG
C104 00         SFLAG   FCB   0         SKIP WRITE FLAG
C105 0000       COUNT   FDB   0         LINE COUNT
C107 11         PROCH   FCB   $11       PROMPT CHAR
C108 07         RTRCH   FCB   $07       RETRY CHAR
C109 04         EOFCH   FCB   $04       END OF FILE CHAR
C10A 01         PEOPT   FCB   1         PARITY ERROR OPTION
C10B 00 00 00 00 EOFST  FCB   $00,0,0,0,0 END OF FILE STBLNK
C10F 00
                        *
                        *
                        * RECEIVE A FILE
C110 BD  32     RESTAR  BSR   PARFIL    PARSE FILE NAME
C112 25  2B             BCS   REEXB     IF NO ERROR
C114 8E  C3E3           LDX   #MESTTL   . PRINT TITLE
C117 BD  43             BSR   STRING
C119 BD  C1ED           JSR   PRBL
C11C BD  41     REPRM   BSR   DISPRM    . REPEAT: DISPLAY PARMS
C11E 8E  C3FB           LDX   #MESCHA   .    PROMPT FOR CHANGES
C121 BD  39             BSR   STRING
C123 BD  C1DE           JSR   UPCHAR    .    GET REPLY
C126 81  4E             CMPA  #'N       . UNTIL 'NO'
C128 27  05             BEQ   REINI
C12A BD  C1F0           JSR   CHAPRM    .    CHANGE PARMS
```

Update to ELEKTRA!

Package #1	Package #2	Package #3
2 MHz 6809 CPU Board Super Floppy Controller OS-9™ w/Edit. Asm. Debugger	2 MHz 6809 CPU Board Super Floppy Controller 4K Humbug™ and Star-Dos™	2 MHz 6809 CPU Board Super Floppy Controller (No software)
$695.00	**$675.00**	**$550.00**

ELEKTRA OS-9™ with Editor, Assembler, and Debugger **$250.00**
ELEKTRA STAR-DOS™ (Adaptation Guide: $50.00) **$75.00**
OS-9™ Super Modem Program by Epstein Associates **$100.00**

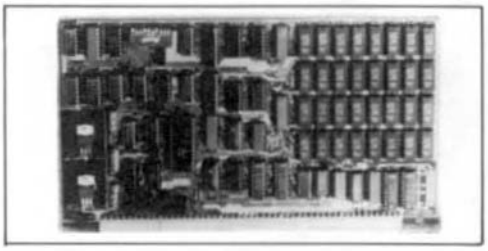
ELEKTRA Super Floppy Controller
(Supports SD, DD, SS, DS, 5", 8", 1MHz, 2MHz, 8 Drives
Emulates the DC-1, DC-2, DC-3, #28, #38, #48, #58 Controllers
Perfect upgrade for the DC-4 **$295.00**

Drivers for TSC's versions of **FLEX™** or **STAR-DOS™** (user installed) **$30.00**
OS-9™ Drivers (Reads and writes CoCo format too, user installed) **$50.00**

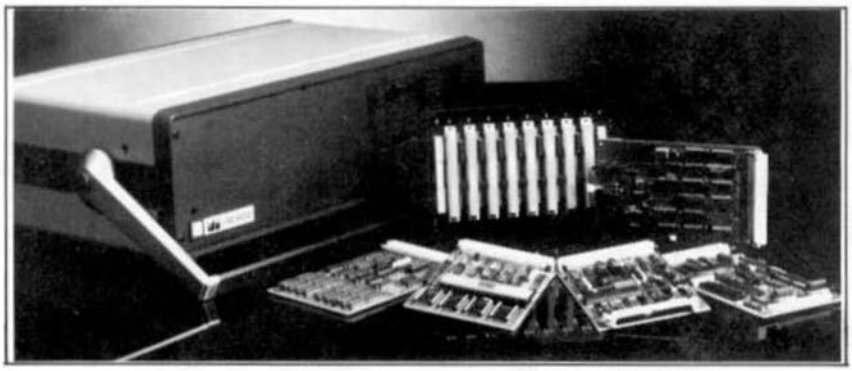
8" Floppy Drive Special w/manual, 90 day warranty
Siemens FDD 100-8 (SSDD) **$135.00**
Siemens FDD 200-8 (DSDD) **$185.00**

Removable Cartridge Winchester Drive (See next page for systems) **$1995.00**

2MHz Memory Boards with on board DAT
by Computer Excellence, Inc.
256K **$749.00** 512K **$1495.00**
1M **$2495.00**



Mizar 68000 — VME Development System with 256K RAM, 360K Floppy, 10 Mbyte Winchester, 4 Serial Ports (synchronous and/or asynchronous), OS-9, Screen Editor, Assembler, "C" compiler, and SASI interface **$6495.00**



Phone:
AAA Chicago Computer Center
Technical Consultation available most weekdays from 4 p.m. to 6 p.m. CST
(312) 459-0450 120 Chestnut Lane Wheeling, IL 60090
See our catalog and ordering information on the next page.

ELEKTRA COMPUTER SYSTEM Includes chassis, dual port serial interface with two cables, CPU 8/9, 4K Humbug, 56K static RAM, super floppy controller with inboard ribbon cable, Star-Dos, dual 80 track DSDD floppy drives (other combinations available; phone). OS-9 may be substituted for HUMBUG and STAR-DOS. $2795.00

ELEKTRA COMPUTER CABINET THE LARGEST SS-50 COMPUTER CABINET AVAILABLE! Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for 'D' type data connectors. Front panel has on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 2 full height or 4 half height 5-1/4" disk drives. $250.00

RACKMOUNT ELEKTRA COMPUTER CABINET 17"w x 21.5"d x 6.7"h. Holds one full or two half height 5-1/4" floppy drives. $250.00

Filler Plate for drive opening: $10.00 **Fan Filter:** $10.00

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 1a @ 16v, 3a @ -16v. Multi-tapped primary for fine tuning. $200.00

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives $50.00. Heavy duty version for 1 Winchester drive and 1 floppy drive or 4 half heights $75.00

AUXILIARY POWER SUPPLY to power second Winchester drive. $125.00

ELEKTRA UNIVERSAL SS-50/SS-50C MOTHERBOARD Heavyweight 0.125" thick 18" long by 9" wide. 11 memory (50 pin) slots. 8 I/O (30 pin) slots. Complete address decoding and selection, as well as extended address capability, for I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. 1" spacing between all memory and I/O slots. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Slow device circuitry permitting 1 Mhz 30 pin disk controllers to run with 2MHz 50 pin CPU boards.
Mounting hardware $5.00 Bareboard w/documentation $80.00
Assembled w/pin connectors $380.00 Assembled w/gold connectors $480.00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested. (Add $25.00 for heavy duty regulator.) $850.00

ELEKTRA 2MHz CPU 8/9 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 2716 EPROMs, 1K scratchpad, and MC6840 triple timer. Run OS-9®, FLEX™, STAR-DOS™. Bareboard $50.00 Assembled $275.00
Optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. $25.00

ELEKTRA DPS DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per port. RTS, CTS, DTR, OCD, IRQ, FIRQ/NMI, and baud rate can be appropriately implemented for each port.
Bareboard $25.00 Assembled $95.00
Cable with jack socket assemblies (two needed per board) Each $25.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request line for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.
Bareboard $25.00 Assembled $80.00
Cable with jack socket assemblies (two needed per board) Each $25.00

ELEKTRA 64K STATIC RAM/ROM MEMORY BOARDS with gold connectors. (In assembled) Assembled and tested. With 56K RAM $269.00 With 64K RAM $299.00

ELEKTRA UNIVERSAL SUPER FLOPPY CONTROLLER THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! Controls up to four 5-1/4" drives and four 8" drives for a total of eight system drives. Single density or double density, 1MHz or 2MHz, 6800 or 6809 (Double density is required for 2MHz). Analog phase locked loop data separators with separate adjustments for 5" and 8" drives. Analog write precompensation circuit with separate adjustments for 5" and 8" drives. Designed to meet the data hold requirements of Western Digital floppy controller IC.
Bareboard (experts only) $100.00 Assembled and tested $295.00
Disk with drivers, setup, and formatting utilities. Specify FLEX 2.0, 6800 Gen FLEX, FLEX 9.0, FLEX 9.1, 6809 Gen FLEX or STAR-DOS, 5" or 8" $30.00
Disk with drivers for OS-9 (Specify 5" or 8") $50.00

ELEKTRA WINCHESTER SYSTEMS THE BEST WINCHESTER SYSTEMS THAT YOU CAN BUY! Has automatic error detection and CORRECTION of up to 11 bit burst errors. SS-50 bus, extended addressing capabilities, DMA, on board sector buffer, drivers included for 6809 FLEX, STAR-DOS, or OS-9. Specify whose version of FLEX that you are using. Drivers for 6800 FLEX2 are available for an additional $100.00. Price includes host interface, controller, drive(s), and cables.
7 Megabyte single drive sys. $1995.00 14 Megabyte dual drive sys. $2995.00
12 Megabyte single drive sys. $2295.00 24 Megabyte dual drive sys. $3595.00
19 Megabyte single drive sys. $2995.00 38 Megabyte dual drive sys. $4695.00
(19 Megabyte drives are the largest that can be supported by FLEX)
6 Megabyte removable cartridge single dr sys. $2995.00 Drive only $1995.00
Circuit boards, cables, software (No drives) 995.00
SS-50C DMA Bus interface board only 695.00

ELEKTRA HD-5 Cabinet for dual 5 1/4" floppy drives with power supply, linecord, fuse power switch, and power cables to drives. 150.00

ELEKTRA HD-5W As above but with EMI filter, fan, and heavy duty power supply. Powers 1 floppy and 1 Winchester or 4 half height 5" floppies. 199.00

5" ribbon cable for dual outboard 5-1/4" disk drives 40.00
2" ribbon cable for dual inboard 5-1/4" disk drives 35.00
Custom cables available Phone

ELEKTRA HD-8 Dual 8" drive cabinet, EMI filter, fan with filter, power supply and power supply cables. 50.00
Filler plate 10.00

8" ribbon cable for dual 8" disk drives 45.00

ELEKTRA 30 PIN PROTOTYPING BOARD 20.00
ELEKTRA 50 PIN PROTOTYPING BOARD 40.00

GOLD 10 PIN CONNECTORS (Specify male with square pins or female) 1.50
TIN 10 PIN CONNECTORS (Specify male with square pins or female) .50

ELEKTRA is a trademark of AAA Chicago Computer Center.
FLEX and **UniFLEX** are trademarks of Technical Systems Consultants, Inc.
HELIX is a trademark of Hazelwood Computer Systems.
HUMBUG, MICROBUG, and STAR-DOS are trademarks of S. A. R. KITS Software Systems Corp.
OS-9 and **BASIC09** are trademarks of Motorola Inc. and Microware Systems Corp.

AAA CHICAGO COMPUTER CENTER (312) 459-0450
120 CHESTNUT LANE • WHEELING, IL 60090
Technical consultation available 4 PM to 6 PM most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 7% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Checks must be drawn on a U.S. bank. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices, specifications, and inventory are subject to change without advance notice.

SUPER MODEM PROGRAM Single character commands. No interrupts required. Transmit manually or transmit disk files (text) of any length to distant computer. Receive and save disk files (text) on local disk system. X-on/X-off supported. Tested for full duplex at speeds up to 9600 baud. Half duplex option. Echo option. Replaces CR with CR/LF (user option). Slow disk file transmit option.
Please specify 6800 or 6809, SSB, STAR-DOS™, or FLEX™, 5" or 8"
Instruction Manual and disk with both source and object code $75.00

OS-9 Super Modem Program by Epstein Associates with autodial, configuration file, etc. 100.00

ALL IN ONE
Editor — Text Processor — Mailing Labels — Mailing Lists — Multiple Form Letters
Use any CRT terminal and printer — Best Package For The Money Anywhere!
Specify 6800 or 6809, SSB, STAR-DOS™, or FLEX™, 5" or 8" 75.00
Add $35.00 for printed source listing; add $100 for source on disk.
All-In-One, Write'n spell, and Spell'n Fix package 250.00

Software by Technical Systems Consultants, Inc.	FLEX™ Source (List)	FLEX™ Source (Disk)	FLEX™ Min. Only	FLEX™ Object w/Man.	UniFLEX™ Add. Man.	UniFLEX™ Man. Only	UniFLEX™ Object w/Man.
Gen. FLEX w/Edit & ASMB	—	—	—	250	—	—	—
FLEX 9.1 (DC-2) w/Edit & ASMB	—	—	5	150	40	100	550
Advanced Programmers Guide	—	—	25	—	—	—	—
Editor	100	250	25	50	—	—	—
Assembler	150	250	25	50	—	—	—
Debug	175	250	25	75	—	—	—
Extended Basic	—	—	25	100	20	50	200
Basic Precompiler	—	—	25	50	10	25	150
Sort/Merge	—	—	25	75	20	35	150
Utilities	—	Inc.	5	75	10	25	150
Diagnostics	—	—	25	75	—	—	—
Text Processor	150	250	25	75	0	35	150
68000 X-ASMB on 6809	—	—	25	250	20	35	300
P scal	—	—	50	200	25	50	300
Rel. ASMB/Linking Loader	—	—	25	150	20	35	175
6800 X-ASMB on 6809	—	—	—	100	—	—	—
Cobol	—	—	—	—	30	75	750
Fortran 77	—	—	—	—	35	65	450

Software by Microware Systems Corp. (Suggested List Prices, varies/mfg)	Run-Time Package	Source	Manual Only	Object w/Man.
OS-9® Level 1 w/Edit, Asm, Debug		400.00	40.00	250.00
OS-9® Level 2 w/Edit, Asm, Debug		400.00	40.00	500.00
OS-9® Edit, Asm, Debug Pkg			25.00	125.00
Device Driver for Disk Controller (Specify Model)		100.00		
Device Driver for ACIA and PIA		50.00		
Clock Driver for 6840 and 58167 clock chips		35.00		
Entertainment Pack 1 or File Handler Toolbox, or NineCom, or Virtual Disk Driver (Level 2 only)			10.00	85.00
Print Spooler (Level 2 only)			15.00	95.00
RMA Relocatable Macro Assembler			20.00	125.00
RMA/68000 Cross Assembler				400.00
BASIC09 W/Run-Time		50.00	N/A	25.00	200.00
BASIC09 Tour Guide Book				18.95
"C" Compiler				25.00	250.00
C Programming Language (Kernighan & Ritchie)				19.95
CIS Cobol Compiler w/Forms 2 Prog. Gen		50.00	N/A	40.00	400.00
Pascal Compiler		50.00	N/A	25.00	250.00
Sage Application Generator		300.00	N/A	25.00	995.00
Microware yearly support service (All products)					150.00
Edition Update w/o manuals 25.00			Version Update w/manuals		75.00

Special Software
STAR-DOS Lt (Specify ELEKTRA or OC-2, or DC-4) $75.00 Adaption guide $50.00
2K MICROBUG 40.00 4K HUMBUG 75.00 Custom versions $85.00
Spell'n Fix by Peter Stark 178.50 Write'n Spell by Peter Stark 75.11
All-In-One, Spell'n Fix, and Write'n Spell package 250.00
SUPER SLEUTH Disassembler System ($101.00 for OS-9 version) 99.00

SD/DD DISK DRIVES	1 head Tandon	2 head s Tandon	2 heads COC	1 head MPI	2 heads MPI
30 day guarantee					
5-1/4", 40 tracks	225.00	300.00	300.00	250.00	325.00
5-1/4", 80 tracks	300.00	375.00	375.00	25.00	400.00
MPI or COC Service Manual (Specify 40 or 80 track)			25.00 Qume DT-8		550.00
Siemens 8" FDD 100-8 (SSDD) $135.00			FDD 200-8 (DSDD)		$185.00

SPECIAL BOARDS
Microtime II Calendar and Clock Board (Assembled) 60.00
Data Mart 16K EPROM bareboard (2708 chips) 30.00

OUTBOARD EPROM PROGRAMMERS BY OPTIMAL TECHNOLOGY
Model EP-2A-79 (Personality modules extra) 189.00
Optimal Technology, Inc. 30 pin parallel I/O board for EP-2A-79 37.00
FLEX™ Software package for EP-2A-79 (Specify 6800 or 6809) 30.00
OS-9 Software package for EP-2A-79 10.00
Model EP-2B-87 (RS-232/20 MA, Motorola Imt, 8K buffer, 1200/9600 baud) 575.00
Model EP-2B-88-4 (Copies 1 to 4 EPROMS) 550.00
Personality Copy Modules for 2708, 2716, 27C16, 2732, 27C32, 2732A, 2758, MCM68764, MCM68766, 2764, 27C64, 2764A, 27128, 27128A, 27256, 27C256, 2508, 2516, 2532, 2564, 25128, 2816, 2816A, R87C32, 8751, 38E70 $17 to $39

Smoke Signal Broadcasting
DCB-4A Double Density Controller Board for 5" and 8" with DOS 549.00
SCB-69 6809 CPU Board 399.00

GIMIX CLEARANCE SALE	LIST	OUR PRICE		LIST	OUR PRICE
Cable (Par I/O)	24.95	20.00	6800 CPU board	24.03	100.00
80 x 24 Video Boards	398.76	250.00	Single prt ser, 1 cable	113.36	90.00
64 X 16 Video Boards	198.71	100.00	Dual prt par, 2 cables	138.32	110.00
16K Mem Bds. w/ ntrl reg		145.00	Systems (trade-ins)		Phone
63L422 DAT chip	17.50	15.00			

HELIX
64K 6809 Computer $2395.00 Other computer systems available
DMA 5" and 8" Floppy Controller 495.00 6809 CPU Board 495.00
68008 board for SS-50 595.00 CP/M-68K 350.00

COMPUTER EXCELLENCE
2MHz Memory board with on board DAT 256K $749.00 512K $1495.00 1M $2495.00

MIZAR
68000, VME development system with 256K RAM, 360K floppy, 10 Mbyte Winchester, 4 serial ports (synchronous and/or asynchronous), OS-9, screen editor, assembler, "C" compiler, and SASI interface $6495.00

SPECIALS
* SSB BFO Floppy Disk Controller (Version 3) Run FLEX or SSB DOS 100.00
* SWTPC DC-4* 230.00 MP-Mb (4K bareboard) 9.95
* SWTPC DMF-2* $95.00 S-32* RAM not included 124.50
* SWTPC MP-09 2MHz CPU* $295.00 *While supplies last
* High speed tape reader 50.00 300 Baud acoustic modem 129.00
* TI 810 Printer w/lower case and full vertical forms control 1200.00

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled boards. AAA Chicago Computer Center does warranty and maintain service for our assembled boards.

```
C128 20 ED          BRA   REFRM     . LOOP
C12F BD C29C REINI  JSR   INITRE    . INITIALIZE
C132 25 0B          BCS   REEND     . IF NO ERROR
C134 BD C31A REGEL  JSR   GLINE     .    REPEAT; GET LINE OF INPUT
C137 25 05          BCS   REFIN     .    UNTIL END OF FILE
C139 BD C385        JSR   PLINE     .       PUT LINE TO FILE
C13C 20 F6          BRA   REGEL     .    LOOP
C13E BD C389 REFIN  JSR   FINIR     .    FINALIZE
C141 7E CD03 REEND  JMP   WARMS     GOTO FLEX
                    *
                    *
                    * PARSE FILE NAME
C144 8E CB40 PARFIL LDX   0FCB      POINT TO FCB
C147 BD CD20        JSR   GETFIL    PICK UP FILE NAME
C14A 25 08          BCS   PAERFI    IF NO ERROR
C14C 86 01          LDA   01
C14E BD C933        JSR   SETEXT    . DEFAULT := .TXT
C151 1C FE          ANDCC 0#FE      . CLEAR FLAG
C153 39             RTS
C154 8E C410 PAERFI LDX   0RESFIL   ELSE
C157 BD 03          BSR   STRING    . REPORT ERROR
C159 1A 01          ORCC  0#01      SET FLAG
C15B 39             RTS           RETURN
                    *
                    *
                    * BEGIN NEW LINE AND PRINT STRING
C15C 7E CD1E STRING JMP   PSTRNG
                    *
                    *
                    * DISPLAY TRANSFER PARAMETERS
C15F BD C1E3 DISPRM JSR   PRNL      BEGIN NEW LINE
C162 8E C432        LDX   0#ESTRA   DISPLAY TITLE
C165 BD F5          BSR   STRING
C167 8E C447        LDX   0#ESPRO   DISPLAY PROMPT CHARACTER
C16A BD 59          BSR   INDENT
C16C 8E C107        LDX   0PROCH
C16F BD 5F          BSR   BYTE
C171 8E C457        LDX   0#ESRET   DISPLAY RETRY CHAR
C174 BD 4F          BSR   INDENT
C176 8E C108        LDX   0RTRCH
C179 BD 53          BSR   BYTE
C17B 79 C109        TST   EOFCH     IF CONTROL CHAR
C17E 2B 0C          BMI   DISTR
C180 8E C466        LDX   0#ESEOF   . DISPLAY END OF FILE CHAR
C183 BD 40          BSR   INDENT

C185 8E C109        LDX   0EOFCH
C188 BD 45          BSR   BYTE
C18A 20 1A          BRA   DIPED     ELSE
C18C 8E C478 DISTR  LDX   0#ESSTR   . DISPLAY END OF FILE STRING
C18F BD 34          BSR   INDENT
C191 86 22          LDA   0'"       . OPEN QUOTE
C193 BD 55          BSR   SCHAR
C195 8E C108        LDX   0EOFST
C198 A6 80 DIPST    LDA   ,X+
C19A 81 0D          CMPA  0#0D
C19C 27 0A          BEQ   DIQUO
C19E BD 4A          BSR   SCHAR
C1A0 20 F6          BRA   DIPST
C1A2 86 22 DIQUO    LDA   0'"       . CLOSE QUOTE
C1A4 BD 44          BSR   SCHAR
C1A6 8E C491 DIPED  LDX   0#ESPED   DISPLAY ERROR OPTION
C1A9 BD 1A          BSR   INDENT
C1AB 8E C4A8        LDX   0#ESACC
C1AE 86 C10A        LDA   PEOPT
C1B1 81 03          CMPA  0#RETRY
C1B3 26 05          BNE   PIMPA
C1B5 8E C4B4        LDX   0#ESRTR
C1B8 20 07          BRA   PIEOC
C1BA 81 02 PIMPA    CMPA  0#ABRT
C1BC 26 03          BNE   PIEOC
C1BE 8E C4AF        LDX   0#ESABR
C1C1 BD 10 PIEOC    BSR   SEGMNT
C1C3 20 2B          BRA   PRNL      BEGIN NEW LINE; RETURN
                    *
                    *
                    * PRINT SII SPACES & I-->STRING
C1C5 34 10 INDENT   PSHS  X
C1C7 8E C4BA        LDX   0#ESSPA
C1CA BD 90          BSR   STRING
C1CC 35 10          PULS  X

C1CE 20 03          BRA   SEGMNT
                    *
                    *
                    * PRINT X-->BYTE IN HEX
C1D0 7E CD3C BYTE   JMP   OUTHEX
                    *
                    *
                    * PRINT X-->STRING
C1D3 A6 80 SEGMNT   LDA   ,X+
C1D5 81 04          CMPA  0#04
C1D7 27 04          BEQ   SERET
C1D9 BD 0F          BSR   SCHAR
C1DB 20 F6          BRA   SEGMNT
C1DD 39             SERET RTS
                    *
                    *
                    * INPUT A CHARACTER IN UPPER CASE
C1DE BD 07 UPCHAR   BSR   ACHAR
C1E0 81 60          CMPA  0#60
C1E2 23 02          BLS   *+4
C1E4 84 DF          ANDA  0#DF
C1E6 39             RTS
                    *
                    *
                    * INPUT A CHARACTER
C1E7 7E CD15 ACHAR  JMP   GETCHR
                    *
                    *
                    * PRINT A CHARACTER
C1EA 7E CD1B SCHAR  JMP   PUTCHR
                    *
                    *
                    * BEGIN NEW LINE
C1ED 7E CD24 PRNL   JMP   PCRLF
                    *
                    *
                    * CHANGE TRANSFER PARAMETERS
C1F0 BD F0 CHAPRM   BSR   PRNL      BEGIN NEW LINE
C1F2 8E C4C1        LDX   0#ESPRE   DISPLAY INSTRUCTION
C1F5 BD 68          BSR   STRINV
C1F7 BD F4          BSR   PRNL      BEGIN NEW LINE
C1F9 8E C447        LDX   0#ESPRO   PROMPT FOR PROMPT CHAR
C1FC BD 67          BSR   CONTR     GET REPLY
C1FE 25 03          BCS   CHRTR     IF NOT NULL
C200 B7 C107        STA   PROCH     . INSERT PROMPT CHAR
C203 8E C457 CHRTR  LDX   0#ESRET   PROMPT FOR RETRY CHAR
C206 BD 58          BSR   CONTR     GET REPLY
C208 25 03          BCS   CHEFT     IF NOT NULL
C20A B7 C108        STA   RTRCH     . INSERT RETRY CHAR
C20D 8E C466 CHEFT  LDX   0#ESEFT   REPEAT; PROMPT FOR EOF TYPE
C210 BD 50          BSR   STRINV
C212 BD CR          BSR   UPCHAR    . GET REPLY
C214 81 0D          CMPA  0#0D
C216 27 2A          BEQ   CHPED     UNTIL CR
C218 81 43          CMPA  0'C
C21A 26 0C          BNE   CHTST     . IF "C"
C21C 8E C466        LDX   0#ESEOF   .    PROMPT FOR EOF CHAR
C21F BD 44          BSR   CONTR     .    GET REPLY
C221 25 1F          BCS   CHPED     UNTIL NULL
C223 B7 C109        STA   EOFCH     .    INSERT EOF CHAR
C226 20 1A          BRA   CHPED     . ELSE
C228 81 53 CHTST    CMPA  0'S
C22A 26 0B          BNE   CHTRU     . IF "S"
C22C 8E C478        LDX   0#ESSTR   .    PROMPT FOR EOF STRING
C22F BD 2C7         JSR   WORD      .    GET REPLY
C232 20 0E          BRA   CHPED     . ELSE
C234 81 4E CHTRU    CMPA  0'N
C236 26 05          BNE   CHEFT     . IF "N"
C238 B6 0B          LDA   0#0B
C23A B7 C108        STA   EOFST     .    INSERT NULL STRING
C23D B6 FF          LDA   0#FF      .    SET TYPE FLAG
C23F B7 C109        STA   EOFCH     UNTIL VALID REPLY
C242 8E C509 CHPED  LDX   0#ESOPT   REPEAT
C245 BD 10          BSR   STRINV    . PROMPT FOR ERROR OPTION
C247 BD 95          BSR   UPCHAR    . GET REPLY
C249 81 0D          CMPA  0#0D
C24B 27 13          BEQ   CHRET     UNTIL NULL
C24D C6 01          LDB   01
C24F 81 41          CMPA  0'A       . IF "A"
C251 27 0A          BEQ   CHSEO     .    OPT := ABORT
```

```
C253 5C                 INCB
C254 81   40            CMPA  #'M       . ELIF "M"
C256 27   05            BEQ   CHSEO     .    OPT := MARK
C258 5C                 INCB
C259 81   52            CMPA  #'R       . ELIF "R"
C25B 26   E3            BNE   CHPEO     .    OPT := RETRY
C25D F7   C10A  CHSED   STB   PEOPT     UNTIL VALID REPLY
C260 20   BB    CHRET   BRA   PRNL      BEGIN NEW LINE;  RETURN
C262 7E   C15C  STRINV  JMP   STRING

                *
                *
                * PROMPT AND HEX BYTE INPUT
C265 1F   12    CONTR   TFR   X,Y
C267 1F   21    COPRO   TFR   Y,X       REPEAT
C269 BD   F7            BSR   STRINV    . PROMPT FOR INPUT
C26B 5F                 CLRB            . COUNT := 0
C26C 34   04            PSHS  B         . RESULT := 0
C26E 34   04            PSHS  B         . ERROR FLAG := 0
C270 BD   C10E  COCHA   JSR   UPCHAR    . REPEAT;  GET A CHAR
C273 81   0D            CMPA  #$0D
C275 27   30            BEQ   COLEX     . UNTIL CHAR = CR
C277 81   CC01          CMPA  TTYDEL
C27A 27   EB            BEQ   COPRO     . UNTIL CHAR = DELETE
C27C 81   CC00          CMPA  TTYBS
C27F 26   0F            BNE   COCOU     .    IF CHAR = BS
C281 5A                 DECB            .       DECR COUNT
C282 2B   E3            BMI   COPRO     . UNTIL COUNT < 0
C284 64   E4            LSR   0,S
C286 A6   61            LDA   1,S       .       SHIFT OUT 1 DIGIT
C288 44                 LSRA
C289 44                 LSRA
C28A 44                 LSRA
C28B 44                 LSRA
C28C A7   61            STA   1,S
C28E 20   ED            BRA   COCHA     .    ELSE
C290 5C         COCOU   INCB            .       INCR COUNT
C291 C1   03            CMPB  #3        .       IF COUNT < 3
C293 2C   B8            BGE   COCHA
C295 8D   1D            BSR   HEXDIG    .          CONVERT CHAR
C297 69   E4            ROL   0,S
C299 68   61            ASL   1,S       .          SHIFT DIGIT TO RESULT
C29B 68   61            ASL   1,S
C29D 68   61            ASL   1,S
C29F 68   61            ASL   1,S
C2A1 AB   61            ADDA  1,S
C2A3 A7   61            STA   1,S
C2A5 20   C9            BRA   COCHA     . LOOP
C2A7 6D   E4    COLEX   TST   0,S
C2A9 26   BC            BNE   COPRO     UNTIL VALID
C2AB A6   61            LDA   1,S       GET RESULT
C2AD 32   62            LEAS  2,S
C2AF C0   01            SUBB  #1        IF NULL:  SET CARRY
C2B1 39                 RTS             RETURN

                *
                *
                * CONVERT CHAR TO HEX DIGIT
C2B2 80   30    HEXDIG  SUBA  #$30
C2B4 2B   0A            BMI   HEXERR
C2B6 81   09            CMPA  #$09
C2B8 23   02            BLS   HECHE
C2BA 80   07            SUBA  #$07
C2BC 81   0F    HECHE   CMPA  #15
C2BE 23   04            BLS   HEVAL
C2C0 4F         HEXERR  CLRA            IF NOT HEX
C2C1 1A   01            ORCC  #$01      . SET CARRY
C2C3 39                 RTS
C2C4 1C   FE    HEVAL   ANDCC #$FE
C2C6 39                 RTS             RETURN

                *
                *
                * PROMPT AND STRING INPUT
C2C7 1F   12    WORD    TFR   X,Y
C2C9 1F   21    WOPRO   TFR   Y,X       REPEAT
C2CB 8D   95            BSR   STRINV    . PROMPT FOR INPUT
C2CD 5F                 CLRB            . COUNT := 0
C2CE BE   C108          LDX   #BEOFST   . POINT TO RESULT
C2D1 8D   C1E7  WOCHA   JSR   ACHAR     . REPEAT;  GET A CHAR
C2D4 81   08            CMPA  #$08
C2D6 27   1A            BEQ   WOLEX     . UNTIL CHAR = CR
C2D8 81   CC01          CMPA  TTYDEL
```

```
C2DB 27   EC            BEQ   WOPRO     . UNTIL CHAR = DELETE
C2DD 81   CC00          CMPA  TTYBS
C2E0 26   07            BNE   WOCOU     .    IF CHAR = BS
C2E2 5A                 DECB            .       DECR COUNT
C2E3 2B   E4            BMI   WOPRO     . UNTIL COUNT < 0
C2E5 30   1F            LEAX  -1,X
C2E7 20   EB            BRA   WOCHA     .    ELSE
C2E9 5C         WOCOU   INCB            .       INCR COUNT
C2EA C1   05            CMPB  #5        .       IF COUNT < 5
C2EC 2C   E3            BGE   WOCHA
C2EE A7   80            STA   ,X+       .          INSERT CHAR
C2F0 20   DF            BRA   WOCHA     . LOOP
C2F2 86   0D    WOLEX   LDA   #$0D      UNTIL VALID INPUT
C2F4 A7   84            STA   0,X       MARK END OF STRING
C2F6 86   FF            LDA   #$FF      SET TYPE FLAG
C2F8 B7   C109          STA   EOFCH
C2FB 39                 RTS             RETURN

                *
                *
                * INITIALIZE FILE RECEIVER
C2FC 7F   C103  INITRE  CLR   PFLAG     CLEAR FLAGS
C2FF 7F   C104          CLR   SFLAG
C302 BD   C1ED          JSR   PRNL      START NEW LINE
C305 8E   CB40          LDX   #FCB      POINT TO FCB
C308 86   02            LDA   #2
C30A A7   84            STA   0,X
C30C BD   C3E0          JSR   FILMS     OPEN FOR WRITE
C30F 26   03            BNE   INEROP    IF ERROR
C311 1C   FE            ANDCC #$FE
C313 39                 RTS
C314 BD   C3DD  INEROP  JSR   ERMES     . REPORT ERROR
C317 1A   01            ORCC  #$01      . SET FLAG
C319 39                 RTS             RETURN

                *
                *
                * GET A LINE OF INPUT INTO LINE BUFFER
           C31A GLINE   EQU   *         REPEAT
C31A 86   3B            LDA   #'*       . SEND "*"
C31C BD   C1EA          JSR   SCHAR
C31F 77   C103          ASR   PFLAG     . IF PREVIOUS PARITY ERROR
C322 24   06            BCC   GLSPB
C324 B6   C108          LDA   RTRCH     .    SEND RETRY CHAR
C327 BD   C1EA          JSR   SCHAR
C32A B6   C107  GLSPR   LDA   PROCH     . SEND PROMPT CHAR
C32D BD   C1EA          JSR   SCHAR
C330 8E   C53E          LDX   #BUFFER
C333 5F                 CLRB
C334 BD   C1E7  GLREP   JSR   ACHAR     . REPEAT;  GET A CHAR
C337 4D                 TSTA            .    IF PARITY ERROR
C338 2A   09            BPL   GLEFC
C33A 34   02            PSHS  A
C33C B6   01            LDA   #$01
C33E B7   C103          STA   PFLAG     .       SET PARITY FLAG
C341 35   02            PULS  A
C343 5D         GLEFC   TSTB            . UNTIL FIRST CHAR = EOF CHAR
C344 26   05            BNE   GLIAP
C346 B1   C109          CMPA  EOFCH
C349 27   34            BEQ   GLSEC
C34B 81   0D    GLIAP   CMPA  #$0D      .    IF CHAR = CR OR LENGTH < LIMIT
C34D 27   04            BEQ   GLAPB
C34F C1   FF            CMPB  #LIMIT
C351 24   E1            BCC   GLREP
C353 A7   80    GLAPB   STA   ,X+       .       APPEND CHAR TO BUFFER
C355 5C                 INCB
C356 81   0D            CMPA  #$0D
C358 26   DA            BNE   GLREP     . UNTIL CHAR = CR
C35A BD   C1ED          JSR   PRNL      . BEGIN NEW LINE
C35D B6   C10A          LDA   PEOPT     UNTIL RETRY NOT REQUIRED
C360 81   03            CMPA  #RETRY
C362 26   05            BNE   GLEFS
C364 7D   C103          TST   PFLAG
C367 26   B1            BNE   GLINE     LOOP
C369 7D   C109  GLEFS   TST   EOFCH     IF STRING USED FOR EOF
C36C 2A   14            BPL   GLRET
C36E BE   C108          LDX   #EOFST
C371 10BE C53E          LDY   #BUFFER
C373 A6   80    GLCOM   LDA   ,X+       . COMPARE STRINGS
C377 A1   A0            CMPA  ,Y+
C379 26   07            BNE   GLRET
C37B 81   0D            CMPA  #$0D
```

```
C37D 26  F6          BNE   GLCOM     IF EQUAL
C37F 1A  01    GLSEC ORCC  #$01      .    SET CARRY
C381 39              RTS             RETURN
C382 1C  FE    GLRET ANDCC #$FE
C384 39              RTS
                     *
                     *
                     * PUT LINE FROM BUFFER TO FILE
C385 7D  C104  PLINE TST   SFLAG     IF SKIP FLAG = NO
C388 26  20          BNE   PLRET
C38A 108E C33E       LDY   #BUFFER
               C38E  PLREP EQU   *       . REPEAT
C38E A6  A0          LDA   ,Y+        .    GET CHAR FROM BUFFER
C390 1F  89          TFR   A,B
C392 84  7F          ANDA  #$7F       .    MASK PARITY
C394 8E  C040        LDX   #FCB
C397 BD  47          JSR   FILMS      .    PUT CHAR TO FILE
C399 26  10          BNE   PLERD      . UNTIL DOS ERROR
C39B 5D              TSTB
C39C 2A  0D          BPL   PLCCR      .    IF PARITY ERROR & OPT = MARK
C39E B6  C10A        LDA   PEOPT
C3A1 81  02          CMPA  #MARK
C3A3 26  06          BNE   PLCCR
C3A5 B6  3F          LDA   #'?
C3A7 BD  37          JSR   FILMS      .       PUT '?' TO FILE
C3A9 26  08          BNE   PLERD      . UNTIL DOS ERROR
C3AB C1  0D    PLCCR CMPB  #$0D       . UNTIL CHAR = CR
C3AD 26  DF          BNE   PLREP      . LOOP
C3AF 8E  C105        LDX   COUNT      . INCR LINE COUNT
C3B2 30  01          LEAX  1,X
C3B4 BF  C105        STX   COUNT
C3B7 39        PLRET RTS             . IF DOS ERROR
C3B8 7C  C104  PLERD INC   SFLAG     .    SET SKIP FLAG; REPORT ERROR
C3BB 20  20          BRA   ERRES      RETURN
                     *
                     *
                     * FINALIZE FILE RECEIVER
C3BD 8E  C040  FINIR LDX   #FCB       POINT TO FCB
C3C0 86  04          LDA   #4
C3C2 A7  84          STA   0,X
C3C4 BD  1A          BSR   FILMS      CLOSE FILE
C3C6 27  02          BEQ   FINSTA     IF ERROR
C3C8 BD  13          BSR   ERRES      . REPORT ERROR
C3CA BD  C1ED  FINSTA JSR  PRNL       BEGIN NEW LINE
C3CD BD  C1ED        JSR   PRNL
C3D0 8E  C105        LDX   #COUNT
C3D3 5F              CLRB
C3D4 BD  CD39        JSR   OUTDEC     DISPLAY LINE COUNT
C3D7 8E  C422        LDX   #MESLIN
C3DA 7E  C1B3        JMP   SEGANT     RETURN
                     *
                     *
                     * REPORT DISK ERROR
C3DD 7E  CD3F  ERRES JMP   RPTERR
                     *
                     *
                     * DOS FILE MANAGEMENT
C3E0 7E  D406  FILMS JMP   FMS
                     *
                     *
                     * MESSAGES
C3E3 50 52 4F 4D MESTTL FCC   'PROMPTING FILE RECEIVER'
C3E7 50 54 49 4E
C3EB 47 20 46 49
C3EF 4C 45 20 52
C3F3 45 43 45 49
C3F7 56 45 52
C3FA 04              FCB   $04
C3FB 4D 41 4B 45 MESCHG FCC   'MAKE CHANGES (Y/N)? '
C3FF 20 43 48 41
C403 4E 47 45 53
C407 20 28 59 2F
C40B 4E 29 3F 20
C40F 04              FCB   $04
C410 49 4E 56 41 MESFIL FCC   'INVALID FILE NAME'
C414 4C 49 44 20
C418 46 49 4C 45
C41C 20 4E 41 4D
C420 45
C421 04              FCB   $04
C422 20 4C 49 4E MESLIN FCC   ' LINES RECEIVED'
C426 45 53 20 52
C42A 45 43 45 49
C42E 56 45 44
C431 04              FCB   $04
C432 54 52 41 4E MESTRA FCC   'TRANSFER PARAMETERS:'
C436 53 46 45 52
C43A 20 50 41 52
C43E 41 4D 45 54
C442 45 52 53 3A
C446 04              FCB   $04
C447 50 52 4F 4D MESPRO FCC   'PROMPT CHAR = '
C44B 50 54 20 43
C44F 48 41 52 20
C453 3D 20 27
C456 04              FCB   $04
C457 52 45 54 52 MESRET FCC   'RETRY CHAR = '
C45B 59 20 43 48
C45F 41 52 20 3D
C463 20 27
C465 04              FCB   $04
C466 45 4E 44 20 MESEOF FCC   'END OF FILE CHAR = '
C46A 4F 46 20 46
C46E 49 4C 45 20
C472 43 48 41 52
C476 20 3D 20 27
C47A 04              FCB   $04
C47B 45 4E 44 20 MESSTR FCC   'END OF FILE STRING = '
C47F 4F 46 20 46
C483 49 4C 45 20
C487 53 54 52 49
C48B 4E 47 20 3D
C48F 20
C490 04              FCB   $04
C491 50 41 52 49 MESPEO FCC   'PARITY ERROR OPTION = '
C495 54 59 20 45
C499 52 52 4F 52
C49D 20 4F 50 54
C4A1 49 4F 4E 20
C4A5 3D 20
C4A7 04              FCB   $04
C4A8 41 43 43 45 MESACC FCC   'ACCEPT'
C4AC 50 54
C4AE 04              FCB   $04
C4AF 4D 41 52 4B MESMAR FCC   'MARK'
C4B3 04              FCB   $04
C4B4 52 45 54 52 MESRTR FCC   'RETRY'
C4B8 59
C4B9 04              FCB   $04
C4BA 20 20 20 20 MESSPA FCC   '     '
C4BE 20 20
C4C0 04              FCB   $04
C4C1 50 52 45 53 MESPRE FCC   'PRESS RETURN TO SKIP'
C4C5 53 20 52 45
C4C9 54 55 52 4E
C4CD 20 54 4F 20
C4D1 53 4B 49 50
C4D5 04              FCB   $04
C4D6 4E 55 4C 4C MESEFT FCC   'NULL (N), STRING (S) OR CHAR (C) '
C4DA 20 28 4E 29
C4DE 2C 20 53 54
C4E2 52 49 4E 47
C4E6 20 28 53 29
C4EA 20 4F 52 20
C4EE 43 48 41 52
C4F2 20 28 43 29
C4F6 20
C4F7 46 4F 52 20          FCC   'FOR END OF FILE? '
C4FB 45 4E 44 20
C4FF 4F 46 20 46
C503 49 4C 45 3F
C507 20
C508 04              FCB   $04
C509 41 43 43 45 MESOPT FCC   'ACCEPT (A), MARK (M) OR RETRY (R) '
C50D 50 54 20 28
C511 41 29 2C 20
C515 4D 41 52 4B
C519 20 28 4D 29
C51D 20 4F 52 20
C521 52 45 54 52
```

```
C525 59 20 28 52
C529 29 20
C52B 46 4F 52 20          FCC    'FOR PARITY ERROR? '
C52F 50 41 52 49
C533 54 59 20 45
C537 52 52 4F 52
C53B 3F 20
C53D 04                   FCB    $04
                  *
                  *
                  * LINE BUFFER
C53E              BUFFER  RMB    LIMIT
          C638    BUFEND  EQU    *
                  *
                  *
                  *
                          END    RECEV

0 ERROR(S) DETECTED


SYMBOL TABLE:

ACCEPT 0001  ACHAR  C1E7  BUFEND C638  BUFFER C53E  BYTE   C150
CHAPRM C1F0  CHEFT  C200  CHPED  C242  CHRET  C260  CHRTR  C263
CHSED  C25B  CHTMU  C234  CNTST  C228  COCHA  C270  COCOU  C290
COLE1  C2A7  CONTR  C265  COPRO  C267  COUNT  C105  DIPED  C1A6
DIPST  C198  DIQUO  C1A2  DISPRA C15F  DISTR  C18C  EOFCH  C109
EOFST  C108  ERMES  C3D0  FCB    C840  FILMS  C3E0  FINER  C3BD
FIWSTA C3CA  FMS    0406  GETCHR CD15  GETFIL CD2D  GLAPB  C353
GLC0M  C375  GLEFC  C343  GLEFS  C369  GLIAP  C34B  GLINE  C31A
GLREP  C334  GLRET  C382  GLSEC  C32F  GLSPA  C32A  HECHE  C2BC
HEMON  C2C0  HEVAL  C2C4  HEXDIG C2B2  INDENT C1C5  INERDP C314
INITRE C2FC  LIMIT  00FF  MARK   0002  MESACC C44B  MESCHA C3FB
MESEFT C4D6  MESEOF C466  MESFIL C410  MESLIN C422  MESMAR C44F
MESOPT C509  MESPED C491  MESPRE C4C1  MESPRO C447  MESRET C457
MESRTR C484  MESSPA C4DA  MESSTR C47B  MESTRA C432  MESTTL C3E3
OUTDEC CD39  OUTHEX CD3C  PAERFI C154  PARFIL C144  PCRLF  CD24
PEOPT  C10A  PFLAG  C103  PIEOC  C1C1  PIMMA  C1BA  PLCCR  C3AB
PLERD  C3B9  PLINE  C3B5  PLREP  C3BE  PLRET  C3B7  PRNL   C1ED
PROCH  C107  PSTRNG CD1E  PUTCHR CD1B  RECEV  C100  REERD  C141
REFIN  C13E  REGEL  C134  RELNE  C12F  REPRA  C11C  RESTAR C11D
RETRY  0003  RPTERR CD3F  ATRCH  C108  SCHAR  C1EA  SEGMNT C1D3
SERET  C10D  SETEXT CD33  SFLAG  C104  STRING C15C  STRING C262
TTYBS  CC00  TTYDEL CC01  UCS    C100  UPCHAR C1DE  WARMS  CD03
WOCHA  C201  WOCOU  C2E9  WOLE1  C2F2  WOPRO  C2C9  WORD   C2C7
```

Listing 3: Demonstration file sender.

```
                  * THIS IS A SAMPLE MODEM INPUT ROUTINE WHICH USES
                  * THE ACIA PARITY ERROR FLAG BIT TO INDICATE A
                  * PARITY ERROR ON RECEIVED DATA.  IT ALSO DETECTS
                  * A MODEM DISCONNECT BY MONITORING THE CTS FLAG BIT.
                  * THE ACIA SHOULD BE INITIALIZED FOR SEVEN BITS
                  * PLUS EVEN OR ODD PARITY.
                  *
                  *
            0000  GETCFL  EQU    *          GET CHAR FROM MODEM
  0000 F6   0000  GLREP   LDB    STAT0      REPEAT;  GET STATUS
*** UNDEFINED SYMBOL
  0003 C5   08            BITB   #$08       UNTIL CTS = 0
  0005 26   00            BNE    GCOV
*** UNDEFINED SYMBOL
  0007 C5   01            BITB   #$01       UNTIL READY
  0009 27   F5            BEQ    GLREP
  000B 1F   98            TFR    B,A
  000D B4   40            ANDA   #$40       ISOLATE PARITY ERROR FLAG
  000F 48                 ASLA              SHIFT TO UPPER BIT
  0010 BA   0000          ORA    DATA0      MERGE CHAR WITH FLAG BIT
*** UNDEFINED SYMBOL
  0013 39                 RTS               RETURN

3 ERROR(S) DETECTED


SYMBOL TABLE:

GETCFL 0000  GLREP 0000
```

K-BASIC

KBASIC™ - Version 1.1

A few months back we told you about a new and unique programming tool - KBASIC. Well, at that time there were some shortcomings that although not fatal, were annoying and certainly improvable. Version 1.1, received recently seems to have rectified most of the original problems. After a few days of trial use KBASIC now appears to be an easy to use, state-of-the-art programming tool. And is not just 'Vaporware', but on hand and ready for delivery.

KBASIC is what we should have had available years ago. It makes programming machine language programs literally a snap. It allows the BASIC programmer (novice or pro) to write applications or control programs in BASIC and debug them in BASIC, then compile them to a stand alone binary program, with no run-time package overhead. Most programs can be **faster and easier written** in BASIC, then when running properly compiled and assembled to machine binary programs with the familiar FLEX .CMD extension or OS-9 binary format. The speed factor is astonishingly greater and the space saved (no BASIC interpreter in memory, saves about 12-20K RAM space). Also the precision of KBASIC is greater than any BASIC I have used. KBASIC prints 2 or 3 more digits (17) than Extended BASIC for expressions resulting in numbers of that size or greater. But the best part of all is, **KBASIC is a lot easier than assembler!** If you can write any kind of program in BASIC then KBASIC is just as **simple**, and not very much different from any full feature ordinary BASIC.

KBASIC is available now for both FLEX™ and OS-9™, from S.E. Media (see catalog this issue). KBASIC comes with OSM™, a complete assembler for either KBASIC compiled programs or your own assembler developed programs. A pure XBASIC™-FLEX version will be available in the near future (differing only in random access I/O and the PRINT USING statements) from the current version, many XBASIC or SWTPC BASIC programs will compile with little or no change with this version. Update policies will be very liberal. So you DO NOT need to wait any longer for a BASIC to machine language compiler. KBASIC IS **IT!!**

Listed below are some improvements of this version (1.1) over the previous version 1.0:

Greater execution speed.
The PARAMETERS$ variable has been added. (*)
Expression have been converted from In-Fix to Post-Fix.
Labels processed in two passes.
Improved optimizing.
Compile time files reduced to less than ten.
The variables may be CROSS-REFERENCED.
The labels may be CROSS-REFERENCED.
The variable list at end of compile shows the the data type.
Larger BASIC programs may be compiled.
HELP messages added.
Auto-Delete option of compiler files after assemble.

The OS-9 and FLEX versions use exactly the same calling syntax. BASIC programs written under the FLEX system may be carried over to OS-9. Or written under OS-9 and moved to FLEX, compiled and executed. See OF™ S.E. Media catalog or **O-Pak™** FHL catalog (very important and useful to all FLEX and OS-9 software developers and users).

If KBASIC has problems understanding your inputted command line a HELP menu is printed on the screen.

Coming Features

BCD-math or BINARY-math will be user selectable. BCD is somewhat slower but gives 16 to 17 digits of precisions and is more accurate to the final digit. BINARY is faster but restricted to about 10-11 digits. But, you will be in command and have the choice, **something no other BASIC now allows!** Plus you have the memory savings and speed of machine language program execution.

For those of you wanting immediate delivery use the **FREE S.E. MEDIA WATTS line** to order your KBASIC and start enjoying real programming ease.

* PARAMETERS$ - This allows the passing of parameters entered on the command line string when the program is called. It returns the complete parameter portion up to but not including the C/R.

```
+++ CUTBAIT.CMD,FISH
OS9: cutbait FISH
PARAMETERS$ is equal to "FISH"
```

SUMMARY OF KBASIC INDEX

I N D E X

BIT BUCKET

microware

DATE: September 20, 1984

SUBJECT: MICROWARE INTRODUCES VERSION OF POPULAR OS-9 OPERATING SYSTEM FOR 68000 BASED SYSTEMS

A new version of the OS-9 operating system for 68000-based computers

has been released by Microware Systems Corporation and is now directly available from a number of 68000 computer manufacturers.

OS-9 is a multiuser/multitasking operating system that provides a Unix-style environment on small, medium size, and real-time systems that can't effectively support Unix. It is compatible with application software written in the C language for Unix System V and BSD 4.1 Unix.

The operating system was originally developed for the 6809 microprocessor as a joint effort by Microware and Motorola. Since its introduction in 1979, OS-9 has become the most popular operating system for the 6809 microprocessor and has been licensed to over 100 computer manufacturers. To date, over 25 manufacturers have licensed the 68000 version.

OS-9 is especially popular in small personal computer, industrial automation and communications applications, mostly because of its very compact and efficient executive. Although it provides essentially all the functions of Unix, the size of the ROMable executive is only about 16K bytes. Unlike Unix, OS-9 works very well on systems with modest amounts of memory and mass storage. For example, Radio Shack's OS-9/6809-based TRS-80 Color Computer gives personal computer users a powerful, entry-level system with a Unix-type environment at a cost of well under $1000.

Microware currently offers two programming languages for OS-9, with others on the way. A complete Unix-compatible C language compiler features ROMable output and symbolic debugging capabilities. A cross-compiler version for Vax systems is also available. The other language is Basic09, an interactive compiler for extended, structured Basic. Microware plans to introduce an ISO standard Pascal and Fortran 77 before the end of this year. A number of third-party suppliers offer a growing library of OS-9 application software.

Another interesting aspect of OS-9 is it's unique, modular design and support for very sophisticated modular software techniques. OS-9 programs (as well as OS-9 itself) are built up from groups of "memory modules". Each module has a unique name and type code. Modules can locate each other by means of system calls, and can be shared simultaneously by several tasks. For example, all programs that use floating point math share a common floating point math module. This is true even if the programs are written in different languages. OS-9's modular environment makes software easier to test and maintain, and also accounts for OS-9's extremely efficient use of memory space.

The operating system itself is also modular, so it can be easily configured and customized by the system integrator or even by individual users. To add support for a new I/O device, it is only necessary to create a new driver module. The driver can be installed without a traditional "sysgen" procedure simply by loading it into memory just like any other program. OS-9 will then automatically "boot it in" system tables even while it is running.

Input/output is based on the traditional Unix-type device independent I/O system with tree-structured directories and full file security. Enhancements to standard Unix I/O include text line-oriented I/O, record locking, and file locking. Another important enhancement is an extremely reliable, logical disk organization that is essentially "crash-proof" and eliminates the fearsome disk recovery problems faced by non-technical Unix users.

The standard OS-9 package includes the operating system, over 50 utility commands, a Motorola-compatible relocatable macro assembler, an interactive debugger, a line editor, and a screen editor. Normally, manufacturers of OS-9 based systems provide a copy of OS-9 pre-installed and ready-to-run versions for their hardware.

For those interested in evaluating OS-9 on their own hardware, Microware offers "PortPacks" which includes a single system license, documentation, sample I/O driver source code, a cross or resident C compiler and macro assembler, plus all other software needed to adapt and install OS-9 on a new target system.

Microware is a software house specializing in system software for Motorola family microprocessors. Founded in 1977, it is one of the oldest system software producers. It's primary business is developing operating systems and programming languages which are distributed under license by computer manufacturers. Microware is headquartered in Des Moines, Iowa and has a subsidiary, Microware Japan Ltd., which is based in suburban Tokyo.

Contact:

Dave West
Assistant Marketing Manager
Microware Systems Corporation
1866 NW 114th Street
Des Moines IA 50322
515-224-1929

```
*  "M2"  MODEM              6 18 84        M.J.KREINIK
*  TERM, MODE                              "PUBLIC DOMAIN"
*
*  FOR USE WITH PROMETHEUS "PRO-MODEM 1200"
*  MODEM RUNS AT 1200 BAUD, USE BUFFERED MODE FOR 300
*  PROGRAM IS HARDWARE SPECIFIC (WAVE-MATE JUPITER 2)
*  AND RUNS FULL-DUPLEX (CHARACTERS ECHOED BY MODEM)
*
*        CAVEAT PROGRAMMER !!!
*        *********************
*
WARMS    EQU   $AD03      FLEX EQUATES
PUTCHR   EQU   $AD18
PSTRNG   EQU   $AD1E
PCRLF    EQU   $AD24
*
STATUS   EQU   $FFC4      ACIA EQUATES
CONTRL   EQU   STATUS
DATA     EQU   $FFC5
CSR      EQU   3          ESCAPE TO FLEX
*
         ORG   $A100
         BRA   START
VERSION  FCB   1
*
TBITS    FCB   %00001001
*                INT=0, RTS=0, 7 BITS, 1 STOP, EVEN, /16
RBITS    FCB   %01001001                  -.0545-
*                INT=0, RTS=1, 7 BITS, 1 STOP, EVEN, /16
*
PROMPT   FDB   $0C0A      FF, LF
         FCC   /READY-/
         FDB   $0D0A      CR, LF
         FCB   4
*
START    LDX   #PROMPT
         JSR   PSTRNG
         LDAB  #%00000011 MASTER RESET
         STAB  CONTRL
*
IN       LDAB  RBITS      SET ACIA REGISTERS
         STAB  CONTRL
INI      LDAB  STATUS
         LSRB             RX DATA FULL?
         BCC   OUT        "FIGURE 8 LOOP"
         LDAA  DATA       GET DATA
         ANDA  #$7F       KILL HIGH BIT
         JSR   PUTCHR
          BRA  INI        LOOK FOR NEXT INPUT CHAR

OUT      LDAB  $FFB0      KBD PIA STATUS (6820)
         BPL   IN         BIT 7=DATA PRESENT
         LDAA  $FFB2      KBD PIA DATA
```

```
         ANDA  #$7F      KILL HIGH BIT
         CMPA  #CSR      CSR, CONTROL "C"
         BEQ   EXIT
FOLD     CMPA  #'x       MODEM WANTS UPPER CASE
         BNI   OUT1      FOR COMMAND MODE
         CMPA  #'z
         BCS   OUT1
         SUBA  #$20
OUT1     LDAB  TBITS
         STAB  CONTRL



OUT2     LDAB  STATUS
         BITB  #2        BIT2, TX DATA FULL
         BEQ   OUT2      WAIT FOR EMPTY
         CMPA  #$0D      CR
         BNE   OUT3
         PSHA
         JSR   PCRLF     DO CR, LF TO SCREEN
         PULA
OUT3     STAA  DATA      SEND TO MODEM
         BRA   IN        TEST FOR INPUT
*
EXIT     JSR   PCRLF
         JMP   WARMS

         END   START
```

Ken Russell
6915 N.E. 163rd
Bothell, Wash. 98011

Computer Publishing Center
68 Micro Journal
5900 Cassandra Smith
P.O. Box 849
Hixon, TN. 37343

Dear Don and Readers,

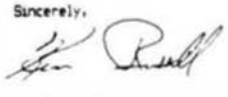
First of all, please note my change of address that appears above, as I sure wouldn't want to miss a single issue of 68 Micro Journal. I thought I might also take this opportunity to share a little program with y'all.

The following listing is a Tektronics 4010 graphics terminal display routine for a Terminus Design, Arcade 50 graphics board. Most of the program is written in FBASIC, the exception is the data file loading routine which was written in assembly language to circumvent some of the string handling limitations of the Basic compiler and also serves to speed up the display. The programs are commented well enough so an explanation shouldn't be necessary here, however, a more detailed discription of the conversion math routines and an explanation of how a Tektronics terminal works can be found in the article in Byte magazine mentioned in the listing.

The display files I have used, were created and downloaded from a mainframe computer (DEC 10 in this case) and are then output to my system via a modem and saved for later display. I had to also write some minor changes to my modem program to allow me to save files without space compression and control character elimination as is normally the case with text files. This is done by merely not setting the space compression flag in Flex's FCB when the received file is being written to disk. I usually also save the files under a .BIN or .TEK extension to save later confusion because listing an output file for a Tektronics, looks like complete gibberish on a normal CRT screen.

Hope there are some out there who have fun with this as it's quite impressive what some of the output can look like even on a lower resolution screen.

Sincerely,



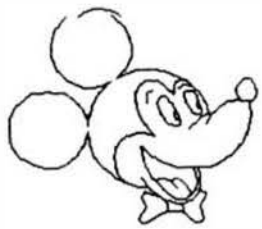
Ken Russell

P.S. Has anyone out there done anything with the MC6839 Floating Point Rom???

```
            OPT 5
REM*************************************************************
REM*          TEKTRONICS DRIVER/EMULATOR                       *
REM*       FOR THE ARCADE 50 GRAPHICS BOARD                    *
REM*       Written in Fbasic by Ken Russell                    *
REM*   TEKFLOAD should be appended to this program             *
REM*     Idea from Byte Magazine (Oct,83 Pg. 439)              *
REM*************************************************************

INIT $E100 :MODE 2          :REM  Init board
GOTO 1000                   :REM  Jump to start

10 REM ** CLEAR SCREEN SUBROUTINE **
   FOR I = $2000 TO $37FF   :REM  Screen memory addr.
    VPOKE I,0
   NEXT I
  RETURN

1000 REM ** MAIN **
  GOSUB 10                  :REM  Clear screen
  M$ = "T"                  :REM  Mode flag. (T = text, B = graphics)
  PRINT "This program will allow you to display a graphics"
  PRINT "output file, intended for a Tek 4010 type graphics"
  PRINT "terminal on an Arcade 50 graphics board."
  PRINT
1005  PRINT
      CALL $2000            :REM  Call data file loader routine
1010  LET A = $3000         :REM  Pointer to data area
1015  C = PEEK(A)           :REM  Get data byte
      IF C = 4 THEN 9990    :REM  4 terminates data field
      IF C <> 29 THEN 1018 :M$ = "B"
1018  IF C <> 31 THEN 1020 :M$ = "T"
1020  IF C = 12 THEN 1025 :GOTO 1030
1025  IF D = 27 GOSUB 10
1030  IF C < 32 THEN 1099
      GOSUB 3000
1099  A = A+1               :REM  Bump data pointer
      LET D = C             :REM  Save last char value
      GOTO 1015

3000  REM  Routine to figure out what is what.
      IF M$ = "T" THEN 4000
      IF C < 64 THEN 3010 :IF C < 96 THEN 3020
      L2 = C :D = C :RETURN
3010  IF D < 96 THEN 3015 :H1 = C :D = C :RETURN
3015  V1 = C :D = C :RETURN
3020  L1 = C :D = C

      REM Actual plotting is done here.
      X = ((H1-32)*32+L1-64)/4 :REM Convert H1 & L1 into X to be plotted
      Y = 192- ((V1-32)*32+L2-96)/4 :REM Convert V1 & L2 into Y.
       IF X > 255 THEN 3030  :REM  Check for out of range
       IF Y > 191 THEN 3035
       GTO 3040
3030  X = 255 :GOTO 3040     :REM  If out of range, clip data
3035  Y = 191 :GOTO 3040

3040  IF M$ <> "A" THEN 3050
       DRAW X,Y :GOTO 3090
3050  MOVE X,Y
3090  M$ = "A"                :REM Set, so next line is draw.
3099  RETURN

4000  VPRINT C
      RETURN

      REM  Exit routine
9990  PRINT: PRINT "Are there any more files to display (Y or N) ";
       INPUT R$
        IF R$ <> "Y" THEN 9999
         GOSUB 10
         GOTO 1005
9999  STOP :END
```

```
*       A PROGRAM TO LOAD A TEKTRONIX COMPATABLE DATA FILE
*       INTO MEMORY AND INTERFACE WITH "TEKDRAW".  THIS
*       FILE SHOULD BE APPENDED WITH TEKDRAW.
*       Written by Ken Russell for Flex9

        LIB     FLEXLIB.TXT         Get Flex equates

DATABF  EQU     $3000       Data buffer beginning

*       Stick command portion in command space

        ORG     $2000

START   PSHS    A,B,X,U     save all registers used here
START1  LDX     #FILMSG     output file name prompt
        JSR     PSTRNG
        JSR     INBUFF      get responce
        LDX     #FCB        now move it to Flex
        JSR     GETFIL
        BCS     START1      if sumthin wrong, try again
        LDA     #1          Function code for Open-Read
        STA     ,X
        JSR     FMS         and go and open it
        BNE     ERROR
        LDA     #$FF        no space compression, because
        STA     59,X        the file looks like binary
        LDA     #0          Function code for Next-Char
        STA     ,X
        LDU     #DATABF     get buffer pointer to put file
LOOP    JSR     FMS         go and get a byte
        BNE     CHKDUN
```

```
        STA     ,U+         save byte
        BRA     LOOP        Keep going 'till done
CHKDUN  LDA     1,X
        CMPA    #$0B        see if end of file
        BNE     ERROR       if not something else wrong
        JSR     FMSCLS      else close the file
        LDA     #4          and mark end of file to
        STA     ,U+         display program
        STA     ,U+
        STA     ,U+
        STA     ,U
        PULS    A,B,X,U,PC  exit back to main
ERROR   JSR     RPTERR
        JSR     FMSCLS
        PULS    A,B,X,U     adjust stack
        JMP     WARMS

FILMSG  FCC     'Enter data file name - INCLUDING EXTENSION :'
        FCB     4

        END
```



"A program always knows the difference between test data and real data...."

old saying

The pictures above, are a screen dump of some Tektronics output files displayed on an Arcade 50 board.

Mr Don Williams,
68 Micro Journal,
5900 Cassandra Smith,
P.O. Box 849,
Hixson, TN 37343,
USA.

Dear Sir,

I recently purchased the PL/9 compiler from Windrush, an excellent product with a superb manual. I did not purchase Mace as I have the TSC Macro Assembler (maybe Windrush should offer a package of PL/9 and Mace), so I had to generate the GEN statements by hand, a laborious and error-prone task. I decided that a computerised solution was required and wrote the program GENPL9.BAS to accomplish this.

The short program is in TSC XBASIC, although as an exercise I should have written it in PL/9! it requires the user to assemble the source code (already debugged!) using the TSC assembler, and the '0' Flex command to reroute the output to a disk file, so as to generate a correctly fielded file as input to GENPL9.BAS. If a different assembler is used, which tabs the fields at different locations, some of the constants will have to be altered. The program assumes that the first 6 characters on the line are 2 spaces plus the address, and these are skipped. The output can be to the terminal, printer, or to a disk file. The disk file can be incorporated into any PL/9 program, either as a LIBrary file or using an editor.

I hope this will be of some use to some of the other PL/9 users.

Yours sincerely,

Dr. L. Placenza.

This is the typical .OUT file produced by the FLEX commands:

+++O.CMD,1.GENTEST.OUT,ASMB.CMD,GENTEST.TXT +B
(i.e. no binary, no symbol table, no FCC/FCB compression, no line numbers).

```
                C0D3  WARMS   EQU     $C0D3
                C840  SYSFCB  EQU     $C840

  C100                        ORG     $C100
  C100 20   01        TEST    BRA     START
  C102 01                     FCB     1
  C103 8E   C840      START   LDX     #SYSFCB
  C106 4F                     CLRA
  C107 5F                     CLRB
  C108 1F   8B                TFR     A,DP
  C10A 10AE 84                LDY     ,X
  C10D ED   22                STD     2,Y
  C10F 7E   C0D3              JMP     WARMS
                              END     TEST

0 ERROR(S) DETECTED
```

When the above .OUT file is run through GENPL9.BAS the following output is produced:

```
                                /* WARMS   EQU
$C0D3           */
                                /* SYSFCB  EQU
$C840           */
                                /*         ORG
$C100           */
    GEN $20,$01;                /* TEST    BRA
START           */
    GEN $01;                    /*         FCB     1
    */
    GEN $8E,$C8,$40;            /* START   LDX
#SYSFCB          */
    GEN $4F;                    /*         CLRA
    */
    GEN $5F;                    /*         CLRB
    */
    GEN $1F,$8B;                /*         TFR     A,DP
    */
    GEN $10,$AE,$84;            /*         LDY     ,X
    */
    GEN $ED,$22;                /*         STD     2,Y
    */
    GEN $7E,$C0,$03;            /*         JMP
WARMS           */
                                /*         END     TEST
    */
```


Dr L.P.L. Placenza
Chemistry Dept.
University of Transkei
Private Bag X5092
Umtata
Republic of Transkei
Southern Af ica

```
100     REM *** GENPL9.BAS
120     REM       generates GEN statements for PL9.
140     REM                 from an assembly file
160     REM       generated by FLEX command line:
180     REM ***O.CMD,FILE.ASM@.CMD,FILE,*BS
200     REM *** eg: GEN $54,$70;     /* PSHS D@,X,Y,U */
220     REM *** Initialise ***
240 GOSUB 820
260     REM *** Ignore null lines ***
280 INPUT LINE #1,A$ : IF A$="" THEN 280
290     REM *** are we done?
300 EM$=INSTR(1,A$,"ERROR(S) DETECTED")
320 IF EM$<>0 THEN 700 : REM all done!
340 D$=0 : B$="" : Y$=12 : B$=G$
350     REM *** M$=mnemonic, A$=op-codes.
360 M$=MID$(A$,22) : A$=MID$(A$,0,12)
380 X$=1
400 IF Y$<=0 THEN 560 : REM passed op-codes.
420 IF X$>LEN(A$) THEN 560
440 SP$=0 : IF MID$(A$,X$,1)=" " THEN SP$=1
460 IF SP$=1 THEN X$=X$+1 : Y$=Y$-X$ : GOTO 400
480 B$=B$+D$+MID$(A$,X$,2)+","
500 D$=1
520 A$=MID$(A$,X$+2) : Y$=Y$-X$+2
540 GOTO 380
550     REM *** replace last comma with semicolon.
560 IF RIGHT$(B$,1)=",", THEN B$=MID$(B$,1,LEN(B$)-1)+SC$
580     REM *** if a blank line or equate then
600     REM *** don't put a GEN without opcodes.
620 IF D$=0 THEN IF M$="" THEN 680
640 IF D$=0 THEN B$=""
660 PRINT #OP$,B$;TAB(30);SL$+M$;TAB(70);SR$
680 GOTO 280
700 IF OP$=2 THEN CLOSE OP$
720 END
740 IF ERR=8 THEN RESUME 700
760 ON ERROR GOTO 0
780 END
800     REM *** Initialisation.
820 ON ERROR GOTO 740
840 SL$="/* " : SR$=" */" : D$="$" : SC$=";"
860 BL$="       " : G$="    GEN "
880     REM *** D$=flag that opcode found if<>0.
900     REM *** Y$=flag for opcodes done if 0, end
920     REM *** Y$=decreasing length of A$.
940 OP$=0 : REM *** output channel.
960 INPUT"File to read ",FR$
980 OPEN OLD FR$ AS 1
1000 INPUT"Output to T(erminal), P(rinter), or D(isk) ",OT$
1020 IF OT$="P" THEN OPEN "O,PRINT" AS 0
1040 IF OT$<>"D" THEN 1080
1060 INPUT"File to write ",FW$ : OPEN NEW FW$ AS 2 : OP$=2
1080 PRINT : PRINT : RETURN
```


** FLASH-Reader Alert-FLASH**

We have recently received complaints concerning the below listed firm, which was a previous advertiser with 68 Micro Journal.

ANALOG MICRO SYSTEMS
5660 Valmont Road
Boulder, Colo

The complaints concern non-shipment or refund of funds to several individuals. Both from overseas and USA customers.

It is our opinion, based on the information we have gathered, that caution should be exercised in dealing with said firm or the below listed individuals, who have represented themselves to be associated with the above named company.

Al Bugay - Vicki Meadows

We have made repeated calls and sent repeated correspondence, in the past months, attempting to locate these individuals, but to no avail. If you should happen to have any information concerning these individuals, or the above named company, please advise me by our WATTS line, as we anticipate this information, and other information being compiled, to be turned over to the Denver, Colorado law agencies and the U.S. Justice department for action.

DMW

- - -

Note: In the above item there has been a delay that should not have occurred. Most of those complaining did not bring it to our attention soon enough. Not all their fault as most are overseas complaints.

Bear in mind that due to the lag between our advertising close out date and the actual date we print and mail the magazine could amount to several months. On top of that, once we receive a complaint we have to go through certain procedures, to insure that there is some merit to the complaints (our legal protection) and that something can be done.

In the above case, if we could contact the responsible parties, and effect a settlement for those complaining, then we would be satisfied. Otherwise, we will follow up until something is done. However, time is of the essence. We need to make a decision and soon, based on if we get a settlement - then we drop it. If we cannot contact these individuals, then we shall PRESS every legal resource to compel local and federal officials (interstate shipping, foreign orders, U.S. State Dept., etc.) to bring this to a conclusion. We have done this in the past and it worked! Therefore, if you have any information concerning the above, PLEASE contact our office immediately.

OMW

- - -

Don Williams
68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, Tn. 37343

Richard Ingram
3630 Tuscarawas Rd.
Beaver, Pa. 15009
(412) 728-3766

Dear Don;

I have the following equipment I wish to donate to a charitable organization. If you know an organization that will accept these items, please have them contact me.

1. SWTPC cheetah 20 amp. pwr. sup., Electra 6808/6809 CPU bd., F&D MDI-1 disk cont., F&D FDS-1 comm. bd., Shugart SA-455 floppy drive, F&D BRB-1 mother bd., SWTPC RP-R eprom programmer, SWTPC parallel bd., 2 Digital Service Design 16K mem. bds..
2. Southeast Micro Systems DDC-16 disk cont.
3. F&D PRB-1 video display
4. F&D CPU-2 w/Percom 6809 adapter
5. F&D AGA-1a video display
6. Digital Research 64K mem. bd.
7. SWTPC 8K mem. bd.
8. Vertern Digital WD1001-55 hard disk cont.
9. Sunny Int'l heavy duty power sup.
10. Conver pwr. sup.
11. Motorola Micro Chrome 68 "TV Bug" computer
12. TSC Flex DOS, Basic, Xbasic(6809)
13. SWTPC Flex 6809

Richard Ingram

Thank you very much

PERIPHERAL TECHNOLOGY

3760 Lower Roswell Rd.
Marietta, Georgia 30067 USA
404/973-0042

PT-69 SINGLE BOARD COMPUTER SYSTEM

- 6809E CPU WITH A 1.0 MHZ CLOCK FREQUENCY
- DOUBLE SIDED, DOUBLE DENSITY FLOPPY DISK CONTROLLER WHICH CAN CONTROL UP TO FOUR 5-1/4" FLOPPY DRIVES
- FLOPPY CONTROLLER USES THE NEW WESTERN DIGITAL WD2797 CONTROLLER CHIP WHICH CONTAINS A HIGH PERFORMANCE DATA SEPARATOR
- (WO RS-232 SERIAL PORTS (6850)
- BAUD RATES ARE JUMPER SELECTABLE
- TWO 8 BIT PARALLEL PORTS (6821)
- TIME OF DAY CLOCK (MC146818) THIS CLOCK KEEPS TRACK OF HOURS, MINUTES, SECONDS, DAY, DATE, MONTH, YEAR AND LEAP YEAR AND HAS BATTERY BACKUP CAPABILITY
- CONTAINS 56K OF USER RAM MEMORY
- CONTAINS 2K/4K OF EPROM USING EITHER A 2716 OR 2732 EPROM
- POWER REQUIREMENTS 5V @ 800 MA +12V @ 20MA -12V @ 20MA
- BOARD SIZE 5-1/2" X 8-1/2"

PT-MON

MONITOR FOR USE WITH THE PT-69. CONTAINS COMMANDS FOR BOOTING FLEX, TESTING MEMORY, MEMORY EXAMINE/CHANGE. THIS MONITOR IS SIMILAR TO S-BUG-E EXCEPT IT DOES NOT CONTAIN ANY DAT ROUTINES

PT-69-K2	COMPLETE KIT - CONTAINS BOARD, PT-MON AND ALL PARTS NEEDED TO COMPLETE THE PT-69	250.95
PT-69A	ASSEMBLED AND TESTED PT-69 BOARD	269.95
PT-69S	PT-69 BOARD, CABINET AND POWER SUPPLY (ASSEMBLED AND TESTED)	349.95
PT-69S2	PT-69 COMPUTER SYSTEM. CONTAINS PT-69 BOARD, TWO DOUBLE SIDED/DOUBLE DENSITY 5-1/4" DRIVES CABINET AND POWER SUPPLY. YOU NEED ONLY TO ADD A CRT AND PRINTER FOR A COMPLETE SYSTEM	699.95
92-80	SAME AS PT-69S2 EXCEPT 80 TRACK DRIVES	1199.95
PR-1	PRINTER FOR PT69. PRINTS 110CPS ON 8-1/2" PAPER INCLUDES INTERFACE FOR PARALLEL PORT	499.95
PR-2	CENTRONICS TYPE PARALLEL INTERFACE FOR PT-69 INCLUDES 4 FOOT CABLE TO PRINTER (INCLUDED WITH PR-1)	49.95
CRT-1	ADDS VIEWPOINT CRT WITH GREEN SCREEN FOR PT-69	549.95
FLEX	FLEX FOR PT-69. INCLUDES EDIT, ASM, FLEX-D AND SOURCE OF PT-MON	159.95
FLEX-D	FLEX DISK DRIVERS AND DISK FORMAT PROGRAM TO MODIFY FLEX OR GENERAL FLEX FOR USE WITH PT-69	29.95
STAR-DOS	OPERATING SYSTEM (COMPATIBLE WITH FLEX)	75.00
OS/9 L1	OPERATING SYSTEM WITH EDIT, ASM, DEBUG	250.00
PT-MON-L	SOURCE LISTING	20.00
PT-MON-D	SOURCE ON DISK	10.00
BOOT-1	BOOT EPROM FOR FLEX AND OS/9	50.00
BOOT-2	BOOT EPROM WHICH BOOTS FLEX ON POWER-UP	25.00

NOTE: IF A CRT OR PRINTER IS ORDERED WITH A SYSTEM, ALL CONFIGURATION IS PERFORMED AND CABLES ARE PROVIDED

OS-9 FOR PERIPHERAL TECHNOLOGY FD-1 OR FD-2

PERIPHERAL TECHNOLOGY CAN SUPPLY OS-9 FOR SEVERAL S-50 BUSS SYSTEMS USING OUR FD-1 OR FD-2 CONTROLLER. LISTED BELOW ARE SOME OF THE COMBINATIONS THAT WE CAN SUPPLY.

DISK CONTROLLERS

FD-1 PERIPHERAL TECHNOLOGY SINGLE SIDED/SINGLE DENSITY
FD-2 PERIPHERAL TECHNOLOGY DOUBLE SIDED/DOUBLE DENSITY

NOTE: THE FD-1 IS COMPATIBLE WITH THE SWTPC DC1 AND DC2
THE FD-2 IS COMPATIBLE WITH THE SWTPC DC4
THE FD-1 MAY BE MODIFIED TO BE COMPATIBLE WITH THE SWTPC DC-3.

CPU TYPES

GIMIX, SSB, SWTPC, SWTPC-809, HELIX, ELECTRA, PERCOM

CLOCK TYPES

SWTPC - MPT
PT146818 PERIPHERAL TECHNOLOGY S-50 CLOCK BOARD
MM5167 CLOCK
MC6840

ACIA TYPES

ACIA - 6850

PIA TYPES

PIA - 6821

STANDARD VERSIONS OF OS-9 L1 ARE AVAILABLE FOR THE FOLLOWING SYSTEMS:

(1) PERIPHERAL TECHNOLOGY PT-69 COMPUTERS
(2) SWTPC COMPUTER WITH FD-1 CONTROLLER AND MP-T TIMER
(3) SWTPC COMPUTER WITH FD-2 CONTROLLER AND MP-T TIMER

OTHER CONFIGURATIONS CAN BE SUPPLIED IF ALL OF THE DEVICE DRIVERS NEEDED ARE LISTED ON THIS SHEET. PRICE FOR CUSTOM VERSIONS IS $350.00 AND REQUIRES THE USER TO FILL OUT A SYSTEM DESCRIPTION SHEET. CONTACT PERIPHERAL TECHNOLOGY FOR QUOTATIONS FOR OTHER CONFIGURATIONS NOT LISTED ON THIS SHEET. HOWEVER ONE REQUIREMENT IS THAT ALL VERSIONS MUST USE OUR FD-1 OR FD-2 DISK CONTROLLER DRIVERS.

PRICE LIST

FLOPPY CONTROLLERS

FD-2	DOUBLE SIDED/DOUBLE DENSITY 5-1/4" FLOPPY DISK CONTROLLER (ASSEMBLED AND TESTED)	149.95
FD-1	DOUBLE SIDED/SINGLE DENSITY 5-1/4" FLOPPY DISK CONTROLLER (ASSEMBLED AND TESTED)	49.95
FD-1B	BARE FD-1 BOARD AND MANUAL	29.95
FD-S	DOUBLE DENSITY DRIVERS AND DISKETTE FORMAT PROGRAM FOR 6800 FLEX ON 5-1/4" DISKETTE	19.95

SYSTEMS

FD-2DSDD	TWO DOUBLE SIDED/DOUBLE DENSITY, 40 TRACK DRIVES, FD-2 CONTROLLER, CABLES, POWER SUPPLY AND CABINET.	749.95
FD-1DSDD	SAME AS FD-2DSDD BUT ONE DRIVE	499.95

CABINETS & POWER SUPPLIES

PS-1	POWER SUPPLY FOR 5-1/4" DRIVE	39.95
CAB-1	CABINET FOR SINGLE 5-1/4" DRIVE	19.95
CAB-2	CABINET FOR DUAL 5-1/4" DRIVES	34.95

PARTS

MLX	THREE 10 PIN FEMALE MOLEX SOCKETS	2.50
HD-1	34 PIN MALE HEADER	3.50
1771-1	FLOPPY DISK CONTROLLER CHIP	14.95
CB-1	SINGLE DRIVE CABLE	19.95
CB-2	DUAL DRIVE CABLE	29.95
CB-3	TRIPLE DRIVE CABLE	39.95

NOTES: 1 - OTHER DRIVE/CONTROLLER COMBINATIONS ARE AVAILABLE. CONTACT PT FOR A QUOTATION.
2 - FLEX IS A TRADEMARK OF TECHNICAL SYSTEMS CONSULTANTS
3 - THE FD-1 WILL ONLY BE AVAILABLE UNTIL CURRENT SUPPLIES ARE EXHAUSTED.

Francis MASSEN
8 Cite Strauss
L-9357 BETTENDORF/ Luxembourg, Europe
tel: 808021

SOME NOTES ON CONNECTING A HX-20 TO A SWTPC COMPUTER

The HX-20 hand-held computer has gained a rather strong acceptance in Europe, especially in Germany, where it is the most sold hand-held machine (state: jan.84). I use the HX-20 as a travel companion or to have a note-book when taking some holidays. Preparing a program on the HX-20 and running it on the SWTPc is quite easy, as the HX-20 has a completely programmable RS-port, which may be adressed from within BASIC. The transmission protocols are choosen by software : baud-rate, number of bit/caracter, parity, number of stop bits, handshaking procedure can be selected by a group of 5 letters. For instance, to list a file through the

serial port to a printer, using 300 baud, 8 bit, no parity, 2 stop bits and no handshaking all that has to be done is to type LIST "COM0:(28N2F)".
A bare-bones physical connection has 3 lines : TX, RX and Ground. The HX-20 uses a 8-pin female DIN-connector at its serial output, and the corresponding male plug may not always be easy to get; the details how the different pins are allocated are well explained in the HX-20 operations manual.

To exchange programs between a SWTPc computer and a HX-20, here is what I do:
On the SWTPc I load and run the MODEM program from Antikainen/Kask, as published in the MICRO68 Jan.83 issue.
I connect the HX-20 to a serial interface normally used by the SW Pc printer.

Sending a program from HX-20 ==> SWTPc
---
* on the SWTPc run the MODEM program, option 'Save file on receive into buffer' (type CTRL-S)
* on the HX-20 type : SAVE "COM0:(28N2F)",A
The A caracter means saving the program in normal ASCII-code and not in a compressed mode. Typing LIST "COM0:(28N2F)" will do the same.
Now the text of the program being transmitted will appear on the SWTPc terminal, while it is loaded into the SWTPc RAM-buffer. After the transmission is over, the file may be written to a FLEX-disk using the CTRL-W command.

It is clear, that the serial interface on the SWTPc must be set to a baud-rate of 300; I have set up my system so that the baud-rates may be choosen through a rotary-switch.

Sending a program from SWTPc ==> HX-20
---
* on the HX-20 : type LOAD "COM0:(28N2F)"
* on the SWTPc : transmit the file from the Flex-disk using the CNTRL-T command of the program MODEM
If you want to exit nicely after the transmission is over, send as a last caracter CNTRL-Z.
With no handshake implemented, I do not recommend a baud-rate higher than 300 (or 1200 for shorter programs); speeding up the transmission will give you very soon a buffer overflow error!

Using the HX-20 in a dialogue-mode

The MODEM program allows a true computer-to-computer dialogue;
however there are 2 difficulties (partially corrected in the Jan.84 issue of MICRO68):

1. the MODEM program does not send an ECHO back to the sender. That means, if you are in the mode "MODEM ON LINE" and you type a text on the HX-20, that text will appear on the SWTPC terminal, but not on the HX-20 screen : you have to type without a visual control of what you are doing.
It is very easy to correct this in the MODEM program:

Look for the interrupt-handler routine INTER in the program MODEM: here is the original text:
```
     INTER  LDB  [MODCR]
            BPL  CHRIN
            LDA  [MODARE]
            ANDA #$7F
```
Change this to:
```
     INTER  LDB  [MODCR] one caracter is received
            BPL  CHRIN   does it come from terminal?
            LDA  [MODARE] no,it comes from modem
            STA  [MODARE] send it back (echo)
            ANDA #$7F
```

2.If the last caracter received by the SWTPc is a <CR>, add one line-feed (LF) so that the following line does not overwrite the preceeding one.
Look for the routine CHRIN on the MODEM program.
Here is the original text of the 19th end following lines:
```
            JSR TEROUT
     OUT    LDB [MODCR]
```
Change this to:
```
            JSR  TEROUT
            CMPA #$0D   CR ?
            BNE  OUT    no
            LDA  #$0A   yes, add one line-feed
            JSR  TEROUT
            LDA  #$0D   reinsert the original CR
     OUT    LDB  [MODCR]
```

Thats all!

On the HX-20 side, things are a bit more complicated; the reason is that Epson was very unfriendly to its customers, refusing to publish anything else but the bare-bones of the entry-points and locations of monitor-routines or RS-buffers. Fortunately, some informations begin to appear. The december issue of the highly recommandable german computer magazine MC published an article from Rolf Grundler who gives a complet dialogue program for the HX-20, and an assembly listing containing the address of the RSGET routine (FF79), which fetches a caracter from the RS-buffer, and that of RSPUT (FF76), which sends a caracter through the RS-port.

Communication-program for the HX-20

```
Author: Rolf Grundler, published in mc, dec.83
* Translated by F.Massen
*
* the error-variable E is located at $A40
* the variable C$ which contains the symbol to emit/receive
* is located at $A42
*
BD FF79 GETCH   JSR RSGET       fetch a caracter from RSbuffer
FE 0A40         LDX #A40        Error variable, address
E7 01           STAB(X),1       Error code
BD 1B           BSR CHECK
27 02           BEQ NORMAL
86 20             LDAA #$20     If no normal caracter, space
FE 0A42 NORMAL  LDX $A42        load address of variable C$
EE 01           LDX (X),1       pointer
A7 00           STAA(X),0       caracter code to C$
39              RTS

FE 0A42 SENDCH  LDX $A42        load address of C$ into X
EE 01             LDX (X),1     load pointer/caracter to send
A6 00           LDAA(X),0       load caracter
BD 06           BSR CHECK
26 03           BNE RTRN1       do not send an illegal symbol
BD FF76          JSR RSPUT      send through RS-port
39      RTRN1   RTS

81 08   CHECK   CMPA #8         DEL?
27 11           BEQ RTRN2
81 0D           CMPA #13        CR?
27 0D           BEQ RTRN2
81 80           CMPA #$80       control valid ASCII
24 07           BCC ILLEG
81 20           CMPA #$20
25 03           BCS ILLEG
5F              CLRB
20 02           BRA RTRN2
C6 01   ILLEG   LDAB #1         code for illegal symbol
39      RTRN2   RTS
```

It is now not a difficult task, to write a BASIC program for the HX-20, containing this routine; I give here the program from the above mentioned author, as published in MC. Modifi-cations to this short, transparent program are easy to make:

DIALOG

```
Author: Rolf Grundler, published in MC, dec.83
commented and translated by F.Massen

180 MEMSET &1A80                    :REM  reserve memory
220 EXEC &HA44: IF E=0 THEN 320     :REM machine code start
230 I$=INKEY$: IF I$="" THEN 220    :REM get one car  ter
240 REM*** one caracter typed in
250 IF I$=E$ THEN CLOSE: END        :REM CNTRL-L to stop
260 IF I$=S$ THEN 290               :REM CNT  -P to look
270 C$=I$: EXEC &HA5A: GOTO 220     :REM send one caracter
280 REM*** show the received text
290 L=CSRLIN: P=POS(0): LINE INPUT "",I$:REM return cursor pos
300 LOCATE P,L,I: GOTO 220
310 REM*** one caracter received
320 IF C$=CR$ THEN PRINT ELSE PRINT C$;:REM print legal car.
330 GOTO 220
340 REM
350 REM*** PREPARE FORMAT
360 REM
370 WIDTH 20,80: SCROLL 9,0: LOCATE 0,1,0
380 PRINT" DIALOGUE WITH THE HX-20"
390 FOR P=0 TO 2000:NEXT: CLS: LOCATE 0,0,1
400 OPEN "I",#1,"COM0:(28N2F)"      :REM set baud rate
```

```
etc
410 E$=CHR$(12):CR$=CHR$(13):S$=CHR$(16):REM control
codes
420 DEFINT E,O,P,V
430 REM*** get the adresses of E and C$
440 C$=":":V=VARPTR(C$):P=&HA42:GOSUB 600
450 E=0   :V=VARPTR(E) :P=&HA40:GOSUB 600
460 REM*** MACHINE CODE (SEE ASS MBLER LISTING)
470 FOR P=&HA44 TO &HA7E : READ Q :POKE P,Q :NEXT
480 GOTO 220
490 REM
500 REM*** here follows the hex code in DATA statements
501 REM*** see assembler listing
....
....
590 REM*** subroutine poke the address
600 Q=INT(V/256): POKE P,Q : POKE P+1,V-256*Q : RETURN
```

Using these 2 programs, MODEM on the SWTPc and DIALOG on the HX-20, makes the communication between these two machines a snap. However, as very often the communication boils down to an exchange of programs from one computer to another, the MODEM program on the SWTPc, together with a SAVE "COM0:...." and a LOAD "COM0:..." on the HX-20 will do it.

\- - -

Enclosed is a patch to CAT to display 3 files info on one line. The patch is for the 6800 FLEX 2.0 system.

It is a simple patch, however it does require an addition of 33 hex bytes.

Type in the listing using EDIT, and name the program CATPATCH. Example:

EDIT CATPATCH

If you do not have RAM at $C000, change the ORG to where RAM is available in your system.

After you have exit EDIT. Assemble the patch:

ASMB CATPATCH,CATPATCH

If no errors are reported by ASMB then append CATPATCH to CAT.CMD:

APPEND 0.CAT.CMD,CATPATCH.BIN,0.DIR.CMD

After the append is completed, a patched CAT is now named DIR on drive 0. Typing DIR will display a screen full of files 3 to a line.

Thank you for a great magazine and keep up the good work.

Robert T. Leong

```
                    OPT     PAG
                * CATPATCH
                * Robert T. Leong    November 30, 1980
                * Last edit November 30, 1980
0001            VER     EQU     1           Version number
                * Patch the FLEX 2.0 CAT.CMD
                * Prints 3 files info on one line
                * colum 0, 25 and 50
C000                    ORG     $C000       System Utility patch area
C000 20 01      PATCH   BRA     PROGRAM
C002 01                 FCB     VER
C003 16         PROGRAM PSH A
C004 37                 PSH B
C005 B6 AC 29           LDA A   COC         Current Ouput Colum
C008 27 0B              BEQ     RETURN
C00A 81 19              CMP A   #25
C00C 2B 0D              BMI     TAB25       Move to next colum
C00E 81 32              CMP A   #50
C010 2B 0D              BMI     TAB50       Move it if smaller
C012 BD AD 24           JSR     PCRLF       If 3 names printed, then next line
C015 CE A8 44   RETURN  LDX     #$A844      Original instruction
C018 33                 PUL B
C019 32                 PUL A
C01A 39                 RTS
C01B C6 19      TAB25   LDA B   #25
C01D 20 02              BRA     TABTO
C01F C6 32      TAB50   LDA B   #50
C021 86 20      TABTO   LDA A   #$20
C023 BD AD 18   TABTOL  JSR     PUTCHR      Print a space
C026 F1 AC 29           CMP B   COC         Tab stop?
C029 26 F8              BNE     TABTOL
C02B 20 EB              BRA     RETURN      All done tabbing
C02D BD AD 24   PATCH2  JSR     PCRLF
C030 CE 4J 58           LDX     #$A158
C033 39                 RTS
                * Equates
AD18            PUTCHR  EQU     $AD18
AD24            PCRLF   EQU     $AD24
AC29            COC     EQU     $AC29       Current Output Colum
A100            UCA     EQU     $A100       Utility Command Area
                * Patch to CAT.CMD
                * Patch to print a blank line
A20A                    ORG     $A20A       Patch main program
A20A BD C0 2D           JSR     PATCH2      Was a LDX #$A368 "Sectors Left"
                * Don't go to newline every time
A24A                    ORG     $A24A
A24A 01                 NOP                 Was a JSR PCRLF
A24B 01                 NOP                 Now is Nops
A24C 01                 NOP
                * Handle colums
A29A                    ORG     $A29A
A29A BD C0 00           JSR     PATCH       Was a LDX #$AB44 (RTS)...
                        END     UCA
NO ERROR(S) DETECTED
```

Don Williams Sr., Publisher
68 MICRO JOURNAL
3900 Cassandra Smith
Computer Publishing Center
P.O. Box 849
Hixson, TN 37343

Dear Mr. Williams:

I recently took delivery on a MX-80 printer with the GRAFTRAX-80 option installed. This printer so impressed me that I decided to pass on the good experience in the form of a short product review. I hope that the attached article is suitable for publication.

Sincerely,

John J. Strunk

John J. Strunk
P.O. Box 2242
Richardson, Texas 75080
Telephone 214-934-4669

EPSON MX-80 PRINTER INTERFACE with SWTPC COMPUTER

By now you have probably heard the rave reviews on the MX-80 printer. Believe me! They are all true. This printer, by my standards, is the best price-performance printer on the market today

The "plain vanilla" version of the MX-8 comes with a Centronics compatible parallel interface, block graphics, and character sets for five different languages. The block graphics are based on a 2x3 matrix in each character space.

Character Space

The five languages are:
- O American English
- O British English
- O French
- O German
- O Katakana (Japanese)

Of these five, only two (Katakana and American English) are full character sets. The rest use the American English character set and substitute the unique characters for little-used characters.

There are three options offered by Epson which deserve special comment. These options are:
- O Serial interface
- O Friction feed
- O GRAFTRAX-80

The serial interface option is an RS232 / current loop compatable interface. It is supplied in one configuration with a 2K byte buffer. There are other configurations available with smaller buffer sizes. However, the 2K buffer is required to use the GRAFTRAX-80 option. This interface can be strapped to operate at speeds up to 19200 bits per second. One disadvantage to the serial interface is that the SWTPC

MP-S card cannot be used due to the lack of a feedback circuit for the BUSY signal.

The friction feed option appears to be a good buy. However, you need to consider that continuous forms, with this option, must be exactly 9.5 inches wide. The pin feed sprockets are not adjustable without removing the friction feed option.

The GRAFTRAX-80 option, however, is a horse of a different color! This option alone is sufficient reason to purchase the MX-80. Three major characteristics make it worth the money. First, an italics character set was included. Second, high resolution graphics (960 dots per line by 72 lines per inch) are available. Third, all twelve print modes may be turned on and off any where in the text. However, you do pay a price for these goodies. The British, French, German, and Katakana character sets are not available with GRAFTRAX-80.

HARDWARE CONSIDERATIONS

Having a computer and operating system which allowes you to send a printer any of the eight bit codes is definitely an advantage. Quite a few of the GRAFTRAX-80 commands require values in the range 0 through 255.

My hardware consists of a SWTPC chassis with a PERCOM SBC/9 cpu card, LFD-400 and DC3 controller cards, 48K memory, and the OS-9 operating system. The system console is a TI-914 VDU terminal driven from a MP-S card. The system printer is a MX-80 with GRAFTRAX-80 driven by a MP-LA parallel interface card.

The parallel interface allowes data transfer rates of up to 1000 characters per second. At first this sounds ludicrous when considering that the maximum print speed is 80 characters per second, but closer examination of the MX-80 commands will show you that many functions require multiple bytes to print one character. The maximum case being when in the 960 dots per line graphics mode. Here approximately 970 bytes are required to print one line of 80 character spaces.

The MX-80 should be connected to the MP-LA parallel interface per the following table.

MX-80 SOCKET PIN	SIGNAL	MP-LA CONNECTOR PIN
1	Strobe	C2
2	Data 1	A0
3	Data 2	A1
4	Data 3	A2
5	Data 4	A3
6	Data 5	A4
7	Data 6	A5
8	Data 7	A6
9	Data 8	A7
10	Acknlg	C1
11	Busy	C1
12	PE	NC
13	SLCT	NC
14	Auto Feed XT	NC
15	NC	NC
16	0V	NC
17	Chassis Gnd	GND
18	NC	NC
31	INIT	NC
32	Error	NC
33	GND	GND
34	NC	NC
35	+5VDC	NC
36	SLCT in	GND

All remaining pins on the MX-80 socket are for returns and should be connected to ground.

SOFTWARE CONSIDERATIONS

PRINTER DRIVER

Most operating systems have parallel printer drivers as standard. Both FLEX from TSC and OS-9 from Microware have parallel drivers which will handle the MX-80 printer. However, if you do not have an operating system which has a parallel driver, I have included the 6809 source for one in listing 1.

You will notice two routines in this listing, one to initialize the port and one to transfer characters. Both routines assume that the MP-LA card is port 7 at address $E01C.

If you have an unmodified SWTPC 02 or earlier mother board you will need to change this address to $801C for port 7.

APPLICATIONS

To take full advantage of the MX-80 printer your higher level language will need to be able to pass to the printer all 256 possible one-byte codes. Most MX-80

command codes are made up of escape sequences. An example:

ESC E or $1B $45

This command causes all following characters to be printed in the emphasized mode.

Take your time in reading the command descriptions in the GRAFTRAX-80 manual.

There are 47 commands available. After this examination you will probably need quite a bit of 'play' time to learn to fully utilize all the capabilities.

HAVE FUN !!

John J. Strunk
P.O. Box 2242
Richardson, Texas 75080

Help

Dear Mr. Williams,

It seems unfortunate to me that the 68XX hobbyists have never united to form a 68XX users group. I'm certain that we have all envied the hobbyists who were able to take advantage of the CP/M users group. That particular group has made available to it's hobbyists a huge amount of software at a low cost. At present, we still have no comparable organization through which we can share our programming efforts. Perhaps we've all been waiting for the other guy to do it and in waiting have noticed the decline in 68XX hobbbyists.

Therefore, I propose to start a 68XX users group and ask that anyone willing to make software available to the group or interested in further information, please contact me. Send a self addressed, stamped en lope to:

Suzanne Taylor
P. O. Box 30107
Honolulu, HI 96820

K-BASIC Update

Dear Don:

K-BASIC is now much improved. Version 1.1 is now ready for release. As you know some of the complaints were that it would only compile small programs, run slower than expected, and had some compile-assembly errors that were mysterious. These problems have been cured.

K-BASIC now has many nice features including better error processing. Here is a quick summary of the improvments:

* The run-time package is much faster
* The PARAMETER$ variable has been added (program [parameter])
* Expressions are converted from IN-FIX to POST-FIX notation
* Two passes to process labels
* Improved optimizing
* The number of files has been reduced to less than ten
* The variables may be CROSS-REFERENCED
* The labels may be CROSS-REFERENCED
* The variables list at the end of compile shows the data type
* Larger BASIC programs may be compiled
* The assembly output files are AUTO-DE[illegible] after assembly
* When in doubt use KS -? for the HELP message

The OS9 and FLEX versions use exactly the same syntax for invoking the compiler on the command line:

KS [-opts] file name [run-time drive/directory]

If K-BASIC has problems understanding the command line then it will display its new HELP menu. The HELP menu shows the command line invocation syntax and a short explaination of the available options.

In the near future we will be adding an optional binary-math package. It will be user selectable between the BCD-math and binary-math packages. The user makes the choice depending on his application. Also the random-access file I/O and PRINT USING statements will be added. This will allow many of the XBASIC-FLEX application software packages to be transported to OS9 with little or no coovereion problems.

Truly yours,

LLOYD I/O
19535 NE GLISAN STREET
PORTLAND, OR 97230

(503) 666 - 1097

Frank Hoffman
President

Classified Advertising

TELETYPE Model 43 PRINTER - with serial (RS232) interface, and full ASCII keyboard. **LIKE NEW** - New cost $1295.00 - ONLY **$799.00** ready to run - Call Tom - Larry - Bob, CPI 615 842-4600

***

Wanted to Buy: Used 6800 and 6809 FLEX software. Commercial and Public Domain. Send description and asking price to: John Current
P.O.Box 273 Honolulu, HI 96859

***

FOR SALE: Smoke Signal SC B-69 CPU board, DCB-4A Drive Control, BFD Drive Control, SWTPC DC-2 Drive Control, 2 Digital Research 64K Static Ram Boards, Electra Universal Mother Board in SWTPC Chassis with new switching P.S. 2 Dual Serial Cards and 1 Dual Parrallel card. All or any. Paul (203) 628-2432. After 6 o'clock p.m. E.S.T.

***

SELLING OUT! Boards, disk drives, and FLEX software for Gimix and Swtpc and other items at LOW PRICES. Send SASE for list to Manfred Peschke, RFD 3, Goffstown, NH 03045.

***

FOR SALE: GIMIX 256K SYSTEM
Now you can have the very finest for a fraction of its $5800 price! Serious inquiries only. Telephone 10PM-midnight Eastern time, Mon-Fri.
Bernard McIlhany, 404/893-2856.

***

For Sale: Motorola 128K Memory Boards, removed from SWPTC S/09 $795.00, SWPTC 8212 Te minals Demostrators $795.00, Hazelwood Dynamic 64K Memory Boards $395.00 Call ask for Tom 615/ 842-4600

***

We need a 68XX users group!! If you agree, send a SASE to:
S. Taylor Box 30107 Honolulu, HI 96820

***

FREE DISKETTE WITH EVERY $50 PURCHASE

TOLL FREE 1-800-338-6800 For Ordering

TELEX 558 414 PVT STH

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

!!! SPECIALS !!!

"Year End CLEARANCE" --- While they Last

FLEX Software ---------

TSC "FLEX Utilities"	was $75.00,	NOW only	$60.00
TSC "Sort Merge"	was $75.00,	NOW only	$60.00
TSC "6809 BASIC"	was $75.00,	NOW only	$60.00
TSC "Extended BASIC"	was $100.00,	NOW only	$85.00
TSC "DeBug"	was $75.00,	NOW only	$60.00
TSC "FLEX Diagnostics"	was $75.00,	NOW only	$60.00
TSC "Text Processing System"	was $75.00,	NOW only	$60.00
TSC "68000 Cross Assembler"	was $250.00,	NOW only	$220.00

LUCIDATA "TEKPAK" (FLEX9, 5 1/4") -- A Pascal implementation of the Tektronix 40xx Terminal Control System, with Pascal SOURCE. The Manual includes a discussion of how to utilize this package in the graphical library in implementing Vector Drawing, Point Plotting, etc., up through Windowing and Clipping concepts.
Normal Price, $100.00: NOW only $85.00

OS-9 Software ------

MICROWARE "OS-9 File Handler Toolbox"; NOW only $70.00
MICROWARE "Relocating Macro Assembler"; NOW only $170.00

!!! NEW NEW NEW !!!

Computer Systems Consultants, Inc.

CRUDEM TELECOMMUNICATIONS PROGRAM
Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "MODEM7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".

FLEX, CCF, OS-9, UniFLEX; with complete Source - $100.00
without Source - $50.00

M-E-M -- TRUE (not Macros) CROSS ASSEMBLERS
Use your 6809-Based Computer System for developing Software for 1802/5, 6800/01/03/11, 6804, 6805, 6809, 6502/3, 8080/5, 8048, 8051, Z-80, and 68000 Systems. Provides the Assembler Language and Listings normally used on the target Systems. Written in "C"; produces Motorola S-Text for machine independence.

FLEX, CCF, OS-9, UniFLEX each - $50.00
any 3 - $100.00
the complete set (including the C Source) - $200.00

LLOYD I/O

X-BASIC -- A "Native Code" BASIC Compiler
Level I X-BASIC supports sequential files, floating point, 3 sizes of integers, string variables, and arrays. The single-pass compiler compiles to Assembly Language Source Code (which may be assembled by the included OSM Assembler, or by the CRASMB Cross Assemblers). Conditional assembly is used to reduce the size of the run time package. (See Review in Oct. '84 Issue of '68' Micro Journal.)

FLEX, CCF, OS-9 Compiler with OSM Assembler - $199.00

OSM -- Extended 6809 Macro Assembler
Provides local labels, Motorola S-records, and Intel Hex records. Also generates OS-9 Memory modules under FLEX, allowing the maintenance of source code programs for both DOS's on one System.

FLEX, CCF, OS-9 $99.00

CRASMB -- 8-Bit Macro Cross Assembler
Same features as OSM, cross-assembles to 6800/2/8, 6801/3, 6804, 6805, 6809, 6811, 6502, 1802, 8048, 8080/5, Z-8, Z-80. Fully supports the target chip's standard mnemonics and addressing modes.

FLEX, CCF, OS-9 full package -- $399.00

CRASMB 16.32 -- Cross Assembler for the 68000
Same features as 8-Bit Cross Assemblers above $249.00

Compusense Ltd.

CRUNCH COBOL -- COBOL Compiler
This COBOL Compiler supports a large subset of ANSII Level I COBOL with many of the useful Level 2 features. Full support of the FLEX File Structures is implemented including Random Files and the ability to process Keyed Files. Large programs can be segmented and linked at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System.

FLEX, CCF Normally $199.00
Special Introductory Price (while they last) -- $99.95

ASSEMBLERS

Southeast Media

ASTRUX09
A "Structured Assembler for the 6809" which requires the TSC Macro Assembler. Allows direct use of structured statements such as IF, ELSE, DO, REPEAT, etc., and provides indented level formatting of the listing so that the structure is apparent. Re. '68' Micro Journal, Sept. '83 (program was called "STASM09"; has been renamed due to conflicts).

A User reports
"... I'm very pleased and am now writing almost exclusively in (ASTRUX09). I've selected it over --- for all future systems development... As (one) of my early evaluations, I rewrote a rather elaborate routine originally done in assembly. Out of the 1000 bytes of code generated, the (ASTRUX09) version used only 20 more bytes than the original. --- could not handle this program since it uses triple-precision fixed point arithmetic... I have a large body of code already written that is incompatible with --- constructs. No problem with (ASTRUX09) and the structure sure helps in understanding the logic!"

F, CCF - $99.95

TSC

Macro Assembler
The FLEX STANDARD Assembler. Special -- F,CCF $35.00

Relocating Assembler w/Linking Loader
Use with many of the C and Pascal Compilers. F,CCF $150.00

Great Plains Comp. Co.

RAMAC
Relocating, Recursive-Macro Assembler and Linking Loader.
F,CCF $120.00; w/Source $240.00

OmegaSoft

SRALL1
Relocating Assembler and Linking Loader
F,CCF $125.00; for One Year Maint., add $50.00

Windrush Micro Systems

MACE, by Graham Trott. F,CCF - $98.00

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

TOLL FREE 1-800-338-6800 SOUTH EAST MEDIA 5900 Cassandra Smith Rd. Hixson, TN 37343 info (615) 842-4601 CoCo OS-9™ FLEX™ SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

DISASSEMBLERS

Computer Systems Consultants

SUPER SLEUTH

Computer Systems Consultants Super Sleuth is a "Time Tested", reliable, PROVEN Disassembler that has gained acceptance through out the SS-50 Bus Community as an extremely POWERFUL, INTERACTIVE, Software Tool. The Super Sleuth Software Package consists of 3 Programs; SLEUTH (the Disassembler), CHGLBL (used to globally Change Labels to a meaningful Name), and XREF (a Cross Reference Generator for Source Code Files). SLEUTH will Disassemble Memory Resident 6809 Code and 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Binary Disk Files. (See Aug. '83 '68' Micro Journal "Color Users Notes" Column for a full Review.)

Color Computer	SS-50 Bus (all w/ Source)
CCD (32K Req'd)	
Obj. Only $49.00	F, $99.00
CCF, Obj. Only $50.00	U, $100.00
CCF, w/Source $99.00	O, $101.00
CCO, Obj. Only $50.00	

ALL Computer Systems Consultants Software
runs on the Color FLEX Systems
ALL in stock
call 800-338-6800
for IMMEDIATE DELIVERY

Computer Systems Center

DYNAMITE +

An "easy to use", powerful Disassembler for Disk Resident 6809 and 6800 Binary Files. Allows the development of a "Control File" of various Program "Boundaries" during successive disassemblies; can use a Label File which automatically replaces a Hex Location with a Label Name; includes an XREF Utility; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX9, Color Computer (for use with Color FLEX Systems), etc. OS-9 Version includes special OS-9 options.

CCF, Obj. Only $100.00		CCO, " "	$150.00
F, " "	$100.00	O, " "	$150.00
	U, " "	$300.00	

COMPILERS and DECOMPILERS

6809 "Structured" Assembly Lang. Compilers

Windrush Micro Systems

PL/9

By Graham Trott. A combination Editor/Compiler/Debugger, all in ONE PACKAGE; provides a totally INTERACTIVE Program Development Cycle. The Single-Pass Compiler supports large Symbol Names; Variable Types; Pointers; Control Structures (similar to 'C' or 'Pascal'); Stack, A-,B-, and D-Register manipulation; etc. The Source-Oriented Trace/Debugger provides Single Stepping, Break-pointing, etc. An excellent Software Development Tool which provides for the maximum utilization of the power of the 6809.

F, CCF - $198.00

Whimsical Developments

WHIMSICAL

Need the Ease of Design and Maintainability of "Structured Programming" AND the Speed and Control of Assembly Language? Then WHIMSICAL was designed for you! This Single Pass, Recursive Descent Compiler provides the tool for developing simple Utilities to MAJOR Systems in Assembly Language. Supports 3 "Lex" Levels which allow one level of Procedure nesting, or more within "Modules". It is easy to develop programs written for other machines since you are working at the Assembly Language level. Features unified, user-defined I/O; produces ROMable, relocatable, recursive, re-entrant Code; Structured style and statements with Procedures and Modules; supports Byte and Double-Byte primitives with 3 types of Integers (up to 32 bit), Char and Boolean, and unlimited sized Arrays (vectors only); Interrupt handling; unlimited length Variable Names; Variable Initialization (defaults to $00); Include "Source File" directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. To quote Ron Anderson in his comments about WHIMSICAL in the Sept. '83 Issue of '68' Micro Journal that, except for the lack of floats, ".... I have to give this one VERY high rating, ...". It is a FAST Compiler which produces FAST Code (his "Primes" Benchmark ran at 9 secs. on a 2 MHz System).

F and CCF - $195.00

'C' COMPILERS

Windrush Micro Systems

C Compiler

By James McCosh. Full featured C Compiler for the FLEX Operating System (lacking ONLY "bit-fields"), including an Assembler. Requires the TSC Relocating Assembler IF the user wishes to implement his own Libraries.

F and CCF - $295.00

Introl

C Compiler

A full-featured C, streamlined for the 6809. Generates very efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, "68" Micro Journal, May '83). Floats, etc.

F, CCF, and O - $375.00
U - $425.00
One Year Maint. - $100.00

PASCAL COMPILERS

TSC

PASCAL Compiler

Native Code Compiler (UCSD Oriented).

F and CCF - $200.00

Lucidata

PASCAL Compiler

P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows linkage to Assembler Code for maximum flexibility.

F and CCF 5" - $190.00
F 8" - $205.00

OmegaSoft

PASCAL Compiler

For the PROFESSIONAL; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long Int. (32 Bit); Dynamic length strings; Interrupt processing, ROM-able, PIC, Re-Entrant Code, etc. POWERFUL! Includes Source for the Symbolic Debugger, Runtime, and several Utilities. Requires a "Motorola Compatible" Relocating Assembler and Linking Loader.

F and CCF - $425.00
One Year Maint. - $100.00

DECOMPILERS

Southeast Media

DUB (A UniFLEX "basic" De-Compiler)

Re-Create a Source Listing from UniFLEX Compiled basic Programs. Easy to Use; works w/ ALL Versions of UniFLEX basic; Output to Disk or Terminal. Time TESTED and PROVEN; SOLID!

U - $219.95

VISA MasterCard

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

* FLEX is a trademark of Technical Systems Consultants
* OS9 is a trademark of Microware

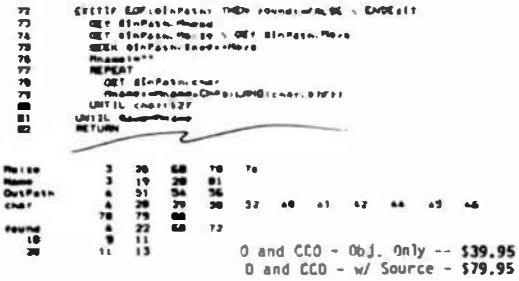
FREE DISKETTE WITH EVERY $50 PURCHASE

UTILITIES

Southeast Media

BasicO9 XRef

This **BasicO9 Cross Reference Utility** is a BasicO9 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also included is a **Program List Utility** which outputs the listing without the overhead of building the cross reference table, whfch allows it to run considerably faster when only a "pretty printed" listing with line numbers is desired. Requires BasicO9 or Run8 for operation.



O and CCO - Obj. Only -- **$39.95**
O and CCO - w/ Source - **$79.95**

Southeast Media

O-F

---- OS/9 to FLEX - FLEX to OS/9 ----

Finally; the barrier has been removed between OS/9 and FLEX formatted disks! Now you can READ **from**, and WRITE **to**, a Single Sided 5" or 8" FLEX diskette from OS-9 with O-F. O-F is a new and unique program, written in BASICO9 (with Source), that performs the following functions;

REFORMAT: A BASICO9 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX.

FLEX: A BASICO9 Program that does the actual read or write function to the special O-F Transfer Disk, all selected from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

O - **$79.95**

Southeast Media

COPYMULT

--- Copy LARGE Disks to several smaller disks ---

The following FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by COPYMULT, a large disk system may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Compile ely documented source files included.

ALL 4 Programs (FLEX, 8" or 5") **$99.50**

Southeast Media

XDATA

A COMMUNICATION Package
for the UniFLEX Operating System

Allows **UniFLEX Based Systems** to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UniFLEX Systems, etc.

-- Verifies Transmission Integrity using checksum or CRC
-- Automatically Re-Transmits bad blocks
-- Transmits data in 128 byte blocks

U - **$299.99**

Southeast Media

OS-9 YDisk

Give your OS-9 Level I System the speed of **memory** access that can be several orders of magnitude over your present floppy disk drive. Use that Extended Memory capability of your SWTPC or Gimix CPU card (or any other that has the same format DAT). The size of the Virtual Disk is completely variable in whole increments of 4K up to 960K, which is all that these systems can address beyond the base page that OS-9 Level I uses. By putting all of your CMDS Directory on your Virtual Disk, you can have the fastest execution speed possible (next to eating up System Memory with all of them). You can also set up high speed inter-process communications via random virtual disk files and not eat up valuable system **memory** with pipe buffers. Some Assembly Required - Level I ONLY.

O, obj. only - **$79.95**
w/ Source - **$149.95**

Lucidata

PASCAL UTILITIES
Requires LUCIDATA Pascal ver 3.

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

F and CCF - **$25.00**

INCLUDE -- allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.

F and CCF - **$25.00**

PROFILER -- produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

F and CCF - **$25.00**

Availability Legends —
F = FLEX, **CCF** = Color Computer FLEX
O = OS-9, **CCO** = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Lucidata

COPYCAT
Pascal NOT required

Allows reading TSC Mini-FLEX, SSB 00S68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in Modular Source Code (Assembly Language) to make it easier to solve unusual problems.

Computer Systems Consultants

FLEX DISK UTILITIES

Eight (8) different FLEX Utilities that should be a part of every FLEX Users Toolbox; Assembly Language (Source Code):

Copy a File with CRC Errors, so it can possibly be salvaged; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order).

PLUS

Ten BASIC Programs to:

A BASIC Resequencer with EXTRAs over "RENUM"; works with ALL versions of FLEX BASIC AND the Precompiler, checks for missing label definitions, processes Disk to Disk instead of in Memory.

Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files.

A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or Source Code). An EXCELLENT Value!

F and CCF - $50.00

BUSINESS

WORD PROCESSING

Windrush Micro Systems

SCREDITOR III

EXTREMELY Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; EXCELLENT Documentation (over 300 pages), including a full Tutorial Section to help you learn how to use the system. Features Cursor-based editing, dynamic Screen Formatting (what you see is what you get), Multi-Column display and editing, "decimal align" columns (AND add them up automatically, if wanted), define multiple keystroke macros, even and odd page number headers and footers, imbed printer control codes in text, full justification series of commands, full "help" support, store common command series on disk for future use, etc. Easy "Set-Up" (for example, you just hit the key you want to use for a specific function, such as "cursor up", and the System reads an stores that key - no digging into tech manuals for codes, etc.); use supplied "set-ups", or remap the keyboard to what you are used too. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

Southeast Media

SPELLB "Computer Dictionary"
OVER 120,000 words!

No more "Let your fingers do the walking through the Dictionary" while you are entering Text with your favorite Editor or Word Processor. SPELLB is more than just "another Spelling Checker"; it allows you to look up a word from within your Editor or Word Processor so that you KNOW it is right WHEN YOU TYPE IT IN with the SPH.CMD Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

F and CCF - $129.95

Great Plains Computer Co.

STYLOGRAPH

A full-screen oriented WORD PROCESSOR -- (now runs on the Data-Comp and FHL Color FLEX Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL CCF - $195.00

F and O - $295.00 U - $395.00

SPELL

Fast Computer Dictionary.

F, CCF, OS/9 - $125.00 U - $175.00

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH.

F, CCF, O - $145.00 U - $195.00

Southeast Media

JUST
Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. JUST is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

- -Output the "Formatted" Text to the Display for format analysis and change.
- -Output the "Formatted" Text to a Text File for use with the supplied FPRINT.CMD for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself).
- -"User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, boldface on and off, etc.
- -Automatic compensation for a "Double Width" printed line.
- -Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
- -Use with ANY Editor.
- -Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

F and CCF - $49.95

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

FREE DISKETTE WITH EVERY $50 PURCHASE

TOLL FREE TELEX 559 414 PVT STH

1-800-338-6800
For Ordering

SOUTH EAST MICRO

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

DATA BASE MANAGEMENT SYSTEMS

Westchester Applied Business Systems
XDMS

Possibly one of the most powerful Database Management Systems' available, this machine language program is small enough to operate on a single sided 5" disk, yet provides the speed of M.L. and **power** limited only by the user's imagination. This DMS supports Relational, Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. **XDMS Level I** provides a functional "entry level" System which provides for defining a Data Base, entering and changing the Data, and producing Reports. **XDMS Level II** adds the POWERFUL "GENERATE" facility which uses an English Language Command Structure in manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. **XDMS Level III** adds several special "Utilities" which provide additional ease of working with the various structures, changing System Parameters, etc.

XDMS Lvl I - F & CCF - $129.95
XDMS Lvl II - F & CCF - $199.95
XDMS Lvl III - F & CCF - $269.95
XDMS System Manual only - $24.95

ACCOUNTING PACKAGES

Great Plains Computer Co. and **Universal Data Research, Inc.** both have Business Packages written in TSC XBASIC for FLEX, CoCo FLEX, and UniFLEX ----

* * * * * * * * * * * * *

- - - - Call 800-338-6800 for more information - - - -

* * * * * * * * * * * * *

Computer Systems Consultants
FULL SCREEN INVENTORY/MRP

The Full Screen Inventory System provides a means of maintaining small inventories. Using a linked, keyed random file structure based upon the item field, it keeps the file in alphabetical order for easier inquiry. With the FIND command, the user may locate and/or print all records matching on partial or complete item, description, vendor, or attributes. Items in backorder or below minimum stock levels may be located and/or printed thru the same process. Printed output may be produced in item or vendor order. A materials requirement planning (MRP) capability for manufacturing environments is included to allow the maintenance and analysis of Hierarchical assemblies of items in the inventory file. It requires TSC's Extended BASIC.

F and CCF - $100.00. U - $150.00

The Virginia Company
BIZPACK

BIZPACK is used for storing accounting, numeric, and financial data which can then be used for planning, budgeting, forecasting, analyzing, etc. While "Electronic Spreadsheets" are extremely useful in many situations, BIZPACK excels in businesses where there are numerous expense columns, revenue sources, significant business indicators, large numbers, erratic week-to-week and month-to-month fluctuations, etc. BIZPACK helps determine statistical relationships, establish trend lines, "smooths" data via moving averages, analyze seasonal data, adjusts for inflation, tags data in Statistics or Column functions, plots data, etc. BIZPACK is oriented toward time series analysis of businesses. The Program displays information on the screen in Columns of information with each Row conforming to a defined Period of Time (weeks, months, years, etc.), and is very easy to use (data is easy to enter, change, and modify; commands can be renamed to suit the users requirements; unlimited ability to create specialized commands using common BASIC Statements; etc.). Requires TSC's Extended BASIC.

F and CCF - $135.00
with Source - $250.00

Computer Systems Consultants
TABULA RASA SPREADSHEET

TABULA RASA is similar to DESKTOP/PLAN and provides for the generation and maintenance of tabular computation schemes often used for analysis of business, sales, and economic scenarios. Its menu-driven user interface provides these capabilities even to those users with no programming experience. Its extensive report-generation capabilities allow the user to generate professional results with minimum effort. It requires TSC's Extended BASIC.

F and CCF - $100.00. U - $200.00

Computer Systems Center
DYNACALC

THE Electronic Spread Sheet for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, a Full Junior College Electronics Curriculum was set up using DYNACALC). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident", Machine Language, this Program is **FAST**. Provides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc. Also available for Data-Comp and FHL FLEX systems using the 50 x 24 Displays.

F and SPECIAL CCF - $200.00
U - $395.00

ODDS AND ENDS

Computer Systems Consultants
FULL SCREEN FORMS DISPLAY

This Package supports any Serial Terminal with cursor control of Memory-Mapped Video Displays. The package substantially extends the screen input/Output capabilities of TSC's Extended BASIC programs by providing a simple, table-driven method of describing and using full screen displays. These table entries are easy to set up and maintain, and are normally stored on disk and read as required. A simple, interactive means of generating the forms and the data field definitions is provided.

F and CCF - $50.00. U - $75.00

Computer Systems Consultants
FULL SCREEN MAILING LIST

The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Using a random fill structure based on the first character of the name field, it maintains the file in alphabetical order for easier inquiry. With the FIND command, the user may locate all records matching on partial or complete name, city, state, zip, or attributes. Printed listings and output to labels may also be produced on the same selective basis. It requires TSC's Extended BASIC.

F and CCF - $100.00. U - $110.00

VISA MasterCard

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

TOLL FREE
1-800-338-6800
For Ordering
SOUTH EAST MICRO
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F - FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Southeast Media

CHESS 6809

Requires FLEX and DISPLAYS On Any Type Terminal

Features:

*Four levels of play.
*Swap side. *Point scoring system.
*Two display boards. *Change skill level.
*Solve Checkmate problems in 1-2-3-4 moves.
*Make move and swap sides. *Play white or black.

This is one of the **strongest** CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

F and CCF - $79.95

Southeast Media

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

F - $59.95
U - $89.95

COLOR COMPUTER SOFTWARE

Stearns Electronics

FORTH

Intrigued by Forth??? Here is a FORTH package **tailored** to the Color Computer! This package is supplied on Tape, with instructions for transferring it to **disk** if you wish. Written primarily in machine language, it's **speed is unparalleled.** A full Semigraphic-8 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in Learning FORTH, a **Trace Feature** is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. (Or; you won't "out grow" the Basic capabilities of this implementation). Combine this package with Leo Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

Color Computer TAPE - $58.95

Custom Software Engineering, Inc.

Color Computer **GRAPHIC SCREEN PRINT** Programs

Dumps any "PMODE" Screen to the Printer with the BASIC USR Function. Shift the Printout Left or Right or Reverse Print (Dark for Light Screen and Vice Versa). All Programs on Tape.

GSPR	for R.S. LP-VII/VIII & DMP 100/200/400	$7.95
GSPRE	for Epson w/ Graftrax and Graftrax +	$9.95
GSPRG	for Gemini 10 and 15	$9.95
GSPRP	for the Prowriter Printers	$9.95

Custom Software Engineering, Inc.

DATE-O-BASE CALENDAR Program

A Menu Driven EXTENDED BASIC Program which allows the entry of up to 12 Memos Per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2099. A **Graphic Calendar** shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR
(Each Tape File will hold up to 400 Memos) $16.95

DISK DATE-O-BASE CALENDAR
(4,000 Memos at 300/Month per Disk) $19.95

Custom Software Engineering, Inc.

That's INTEREST-ing

Interested in INTEREST (the Money Kind)? An EXTENDED BASIC Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Schedules, etc.

TAPE - $29.95

Custom Software Engineering, Inc.

DISK DATA HANDLER 64K

An EXTENDED BASIC Data Management System w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the first; 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of Fields, etc. An extremely POWERFUL TOOL; instructions provide examples of Mailing Lists and a Financial Stock Profit and Loss Tracking System.

DISK - $54.95

Custom Software Engineering, Inc.

DISK DOUBLE ENTRY

DISK EXTENDED BASIC Accounting Program w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Business, Clubs, Churches, Personal Use, etc. Up to four levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. DOE allows up to 300 accounts and a Trial Balance of $9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the journal transaction, which may include comments. Screen reports allow review of past transactions and current balances.

DISK - $44.95

!!! WANTED - ALIVE !!!

Accounts Receivable, Accounts Payable,
Payroll, General Ledger, Inventory

If you have functioning software, give us a
CALL at 1-800-338 6800.

VISA

MasterCard

*FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. What is an electronic spread-sheet, anyway?
Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. Is DYNACALC just for accountants, then?
Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. What will DYNACALC do for ME?
That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. Do I have to learn computer programming?
NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. Do I have to modify my system to use DYNACALC?
Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. Will DYNACALC read my existing data files?
You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. How fast is DYNACALC?
Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. Is there a version of DYNACALC for MY system?
Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. How much does DYNACALC cost?
The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. Where do I order DYNACALC?
See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

UniFLEX software prices include maintenance for the first year.

DYNACALC is a trademark of Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

WINDRUSH MICRO SYSTEMS

UPROM II

PROGRAMS and VERIFIES: 2758, 2508, 2716, 2516, 2732/2732A, MCM68704/6, 2764/2764A, 2564A, 27128/27128A, and 27256. (=Intel, TI=Texas, M=Motorola.

NO PERSONALITY MODULES REQUIRED!

FAI-VOLT EPROMS ARE NOT SUPPORTED

INTEL's intelligent programming (tm) implemented for Intel 2764, 27128 and 27256 devices. Intelligent programming reduces the average programming time of a 2764 from 7 minutes to 1 minute 15 seconds (under FLEX) with greatly improved reliability.

Fully enclosed pod with 3' of flat ribbon cable for connection to the host computer MC6821 PIA interface board.

MC6809 software for FLEX and OS9 (Level 1 or 2, Version 1.2).

BINARY disk FILE offset loader supplied with FLEX, MDOS and OS9.

Menu driven software provides the following facilities:

a. FILL ..... a selected area of the buffer with a HEX char.
b. MOVE ..... blocks of data.
c. DUMP ..... the buffer in HEX and ASCII.
d. FIND ..... a string of bytes in the buffer.
e. EXAMINE/CHANGE . the contents of the buffer.
f. CRC ..... checksum a selected area of the buffer.
g. COPY ..... a selected area of an EPROM into the buffer.
h. VERIFY ..... a selected area of an EPROM against the buffer.
i. PROGRAM ..... a selected area of an EPROM with data in the buffer.
j. SELECT ..... a new EPROM type (return to types menu).
k. ENTER ..... the system monitor.
l. RETURN ..... to the operating system.
m. EXECUTE ..... any DOS utility (only in FLEX and OS9 versions).

FLEX AND OS9 VERSIONS AVAILABLE FROM GIMIX. SSB/MDOS CONTACT US DIRECT.

MACE/XMACE/ASM05

All of these products feature a highly productive environment where the editor and the assembler reside in memory together. Gone are the days of tedius disk load and save operations while you are debugging your code.

- Friendly inter-active environment where you have instant access to the Editor and the Assembler, FLEX utilities and your system monitor.
- MACE can also produce ASMPROCe (GEN statements) for PL/9 with the assembly language source passed to the output as comments.
- XMACE is a cross assembler for the 6800/1/2/3/8 and supports the extended mnemonics of the 6303.
- ASM05 is a cross assembler for the 6805.

D-BUG

LOOKING for a single step tracer and mini in-line disassembler that is easy to use?? Look no further, you have found it. This package is ideal for those small assembly language program debugging sessions. D-BUG occupies less than 6K (including its stack and variables) and may be loaded anywhere in memory. All you do is LOAD IT, AIM IT and GO! (80 col VDUs only).

McCOSH 'C'

This is a complete a 'C' compiler as you will find on any operating system for the 6809. It is completely compatible with UNIX V7 and only lacks 'bit-fields' (which are of little practical use in an 8-bit world!).

- Produces very efficent assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%.
- Supports interleaved assembly language programs.
- INCLUDES its own assembler. The TSC relocating assembler is only required if you want to generate your own libraries.
- The pre-processor, compiler, optimizer, assembler and loader all run independently or under the 'CC' executive. 'CC' makes compiling a program to executable object as simple as typing in 'CC,HELLO.C <RETURN>'.

PL/9

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-debugger, which, amongst other things, can single step the program a SOURCE line at a time. You also have direct access to any FLEX utility and your system monitor.
- 375+ page manual organized as a tutorial with plenty of examples.
- Fast SINGLE PASS compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files.
- Signed and unsigned BYTES and INTEGERS, 32-bit floating point REALs.
- Vectors (single dimension arrays) and pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (\), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/XOR)
- Control statements: IF..THEN..ELSE, EF..CASE1..CASE2..ELSE, BEGIN..END, WHILE... REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (XREG), (CCR) and (STACK).
- FULLY supports the MC6809 RESET, NMI, FIRQ, IRQ, SWI, SWI2, and SWI3 vectors. Writing a self-starting (from Power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Machine code may be embedded in the program via the 'GEN' statement. This enables you to code critical routines in assembly language and embed them in the PL/9 program (see 'MACE' for details).
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library procedure modules are supplied: IOSUBS, BITIO, HARDIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, BASTRING, and REALCON.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

quoted from Ron Anderson's FLEX User Notes column in '68. Need we say more?

IEEE-488

- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:
  - Talker - Serial Poll - Single or Dual Primary Address
  - Listener - Parallel Poll - Secondary Address
  - System controller - Group trigger - Talk only ... Listen only
- Fully documented with a complete reprint of the MILGRAM® article on the IEEE bus and the Motorola publication 'Getting aboard the IEEE Bus'.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. A complete back to back test program is also supplied in the form of a listing. These drivers have been extensively tested and are GUARANTEED to work.
- Single S-30 board (4, 8 or 16 addresses per port), fully socketed, gold plated bus connectors and IEEE interface cable assembly.

PRICES

D-BUG	(6809 FLEX only)	$ 75.00
MACE	(6809 FLEX only)	$ 75.00
XMACE	(6809 FLEX only)	$ 98.00
ASM05	(6809 FLEX only)	$ 98.00
PL/9	(6809 FLEX only)	$198.00
'C'	(6809 FLEX only)	$295.00
IEEE-488	with IEEE-488 cable assembly	$298.00
UPROM-II/U	with one version of software (no cable or interface)	$305.00
UPROM-II/E	as above but complete with cable and S-30 interface	$565.00
CABLE	3' twist-n-flat 50 way cable with IDC connectors	$ 35.00
S-30 INT	S5-30 interface for UPROM-II	$130.00
EXOR INT	Motorola EXORbus (EXORciser) interface for UPROM-II	$195.00
UPROM SFT	Software drivers for 2nd operating system. Specify FLEX or OS9 AND disk size	$ 35.00
UPROM SRC	Assembly language source (contact us direct)	

ALL PRICES INCLUDE AIR MAIL POSTAGE

Terms: CWO. Payment by Int'l Money Order. VISA or MASTER-CARD also accepted.

WORSTEAD LABORATORIES, NORTH WALSHAM, NORFOLK, ENGLAND. NR28 9SA.

TEL: 44 (692) 404086
TLX: 975548 WMICRO G

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, LLOYD I/O, & ALFORD & ASSOCIATES.

FLEX (tm) is a tradeemark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporation, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

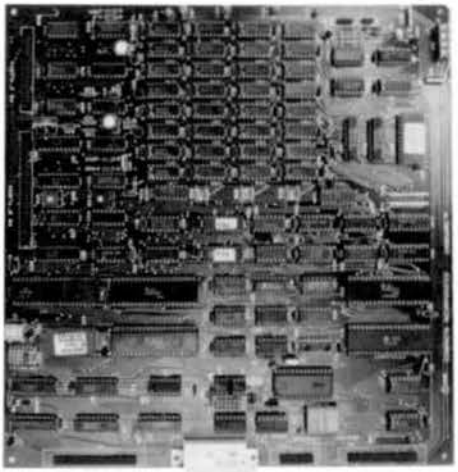
THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!

New! *Lower Price!*



BLANK PC BOARD
$99.95
WITH PAL'S, AND TWO EPROMS.

FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.

$249.00
COMPLETE KIT! FULLY SOCKETED.

PRICE CUT!!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power is available at a truly unbelievably low price.

FEATURES:

- ★ 64K RAM using 4116 RAMS.
- ★ 6809E Motorola CPU.
- ★ Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
- ★ On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- ★ ASCII keyboard parallel input interface. (6522)
- ★ Serial I/O (6551) for RS232C or 20 MA loop.
- ★ Centronics compatible parallel printer interface. (6522)
- ★ Buss expansion interface with DMA channel. (6844)
- ★ Dual timer for real time clock application.
- ★ Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$99
OS9™ from Microware	$199

Specify 5-1/4 or 8 inch

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping. USA AND CANADA ONLY

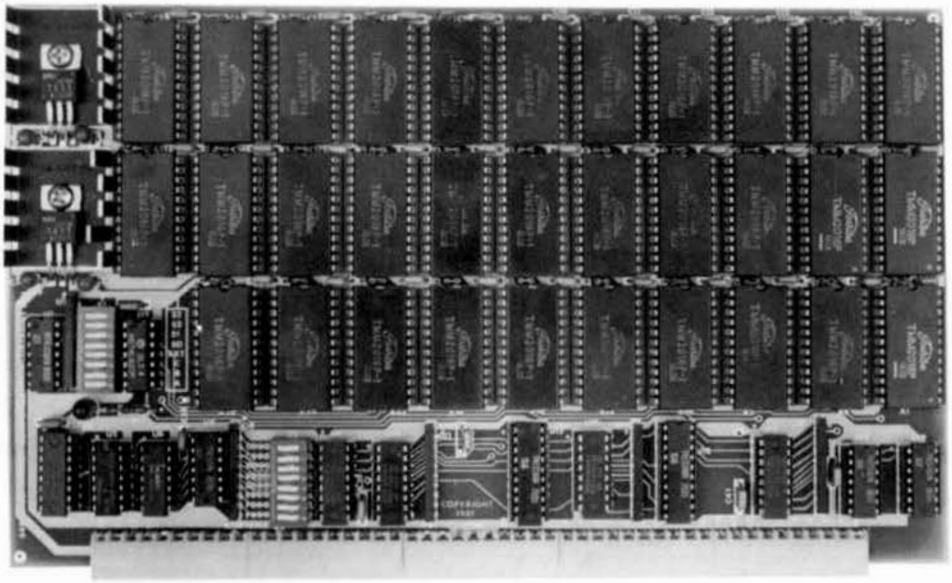
64K SS-50 STATIC RAM

PRICE CUT!!

$149.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $45

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$169
64K	$199

CLOSE OUT SPECIAL
WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

!!! FREE !!!

Published Monthly by Computer Publishing Inc., Hixson, TN.

$1.95

Bulk Rate
U.S. Postage
PAID
Chattanooga, TN
Permit No. 357

Color Micro Journal

The Color Computer Monthly Magazine

$1.95 per issue Vol. 1, Issue 2 October, 1983

THIS 'N THAT

The BIG NEWS this month is that OS-9 has finally arrived for the Color Computer. The ASTOUNDING part of the Radio Shack OS-9 package, besides the price, is the [illegible]. You 'Old Time Radio Shack Followers' will not believe what you see. Jon Shirley has been telling us that the main reason for the "lack" of documentation with a lot of their products was the restrictions placed on releasing that information by MICROSOFT: I

OS-9 on the COLOR COMPUTER

One of the "Operating Systems of the Future" is now available for the "little old Color Computer": OS-9. Freely translated, OS-9 means "Operating System for the 6809" (OS-9 is now being written for the 68000, also). Since it is fairly obvious that UNIX and "UNIX-Type" Operating Systems will be running on just about every computer to come out in the next few years, a whole new language is beginning to appear on the horizon.

Color Computer OS-9; the Package

We had been running a preliminary release of OS-9 on the Color Computer for a few weeks, and received the "Official Radio Shack" version for Review a couple of days ago. To put it mildly, this package is IMPRESSIVE! For $69.95 (Radio Shack Catalog Number 26-3039), you receive a 9 1/2" x 7 5/8" x 2" package containing 4

Introducing NuBASE: the *uncomplicated* data base

It lets you throw away all the books!

NuBASE is a data base manager so versatile that you can use it to do what you want with your data. It's not complicated or overbearing; in fact, it's so easy to use that you'll be up and running in minutes.

Simple user-specified masks insure data accuracy. Data integrity is assured through the use of highly crash-resistant software. *See what you're doing* through the interactive generation of files, screens and reports.

NuBASE is as affordable as it is complete. There's nothing else to buy... $150 brings you the comprehensive package, including a ready-to-use mailing list application to get your **NuBASE** working for you on day one.

The computing power of NuBASE is limited only by the capacity of your hardware.

DO WE HAVE YOUR NAME & ADDRESS
For new products news & announcements?

VISA MasterCard

Currently available for OS-9 Level II
For more information or to place an order, contact:

Dept. 68 16

the JBM group

The JBM Group, Inc.
Continental Business Center
Front & Ford Streets
Bridgeport, PA USA 19405
TWX: 510-660-3999

215-275-1777

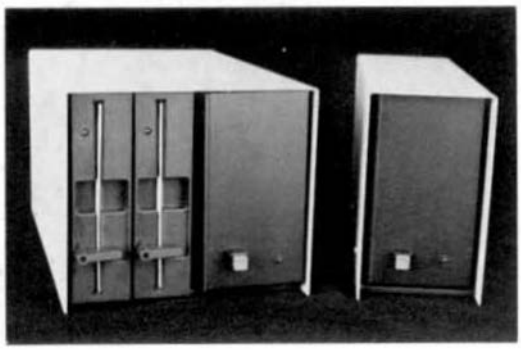
PA res. add 6% sales tax.
US orders, add $5.00 postage and handling.

* OS9 is a registered trademark of Microware Corp.

68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

PT69 SINGLE BOARD COMPUTER SYSTEM
OS-9 NOW AVAILABLE



Pictured:
System with Drives/System without Drives

The proven PT69 Single Board Computer now features OS-9 capability. Powerful performance, reliability, • OS-9 — UNBEATABLE! The PT69 is a complete system in a compact package.

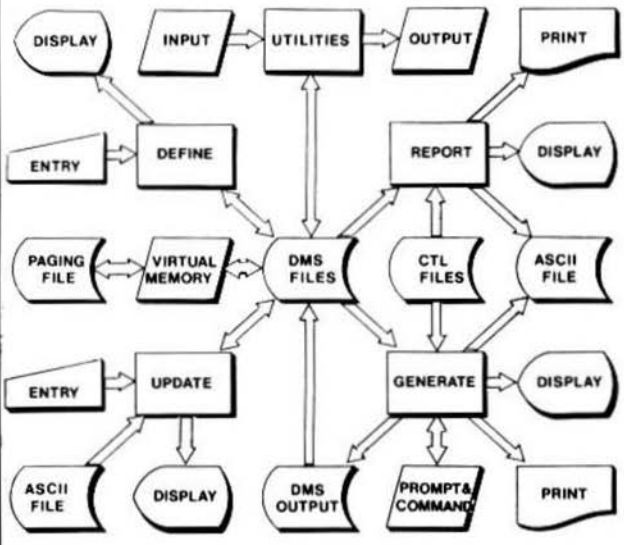
- 1 MHZ 6809E Processor
- 2 RS232 Serial Ports (6850)
- 2 8-Bit Parallel Ports (6821)
- 56K RAM, 4K EPROM
- Time-of-Day Clock (MC146818)

* COMPLETE SYSTEM with PT69 Board, 2 DS/DD 5¼" 40 Track Drives, Cabinet, and Power Supply	$999.95
* PT 69 Board, Assembled and Tested, with Power Supply • Cabinet	$399.95
* PT69, Assembled and Tested Board	$299.95
* Parallel Printer interface with cables	$ 49.95
* OS-9 L1, includes edit, asm, • debug	$250.00
* STAR-DOS Level 1 (Compatible with Flex)	$ 75.00

PERIPHERAL TECHNOLOGY
"Supplying Your Computer Needs Since 1978"
3670 Lower Roswell Road
Marietta, Georgia 30067
VISA/MASTERCARD/CHECK/COD 404/973-0042

*OS-9 is a trademark of Microwere and Motorola
*FLEX is a trademark of Technical Systems Consultants

XDMS
Data Management System



System Architecture

WESTCHESTER Applied Business Systems
Post Office Box 197
Briarcliff Manor, N.Y. 10510

XDMS Data Management System

The XDMS Data Management System is available in three levels. Each level includes the XDMS nucleus, VNDEX utility and System Documentation for level III. XDMS is one of the most powerful systems available for 6809 computers and may be used for a wide variety of applications. XDMS users are registered in our database to permit distribution of product announcements and validation of user upgrades and maintenance requests.

XDMS Level I

XDMS Level I consists of DEFINE, UPDATE and REPORT facilities. This level is intended as an "entry level" system, and permits entry and reporting of data on a "tabular" basis. The REPORT facility supports record and field selection, field merge, sorting, line calculations, column totals and report titling. Control is via a English-like language which is upward compatible with level II. XDMS Level I $129.95

XDMS Level II

Level II adds to Level I the powerful GENERATE facility. This facility can be thought of as a general file processor which can produce reports, forms and form letters as well as file output which may be re-input to the facility. GENERATE may be used in complex processing applications and is controlled by a English-like command language which encompasses that used by Level I. XDMS Level II $169.95

XDMS Level III

Level III includes all of level II plus a set of useful DMS Utilities. These utilities are designed to aid in the development and maintenance of user applications and permit modification of XDMS system parameters, input and output of XDMS files, display and modification of file formats, graphic display of numerical data and other functions. Level III is intended for advanced XDMS users. XDMS Level III $269.95
XDMS System Documentation only ($10, credit toward purchase). . $ 24.95

XACC Accounting System

The XACC General Accounting System is designed for small business environments of up to 10,000 accounts and inventory items. The system integrates accounting functions and inventory plus the general ledger, accounts receivable and payable functions normally sold separately in other systems. Features user defined accounts, products (or services), transactions, invoicing, etc. Easily configured to most environments. XACC General Accounting System (Requires XDMS, pref. Lv. III) . $299.95
XACC System Documentation only ($10, credit toward purchase). . $ 24.95

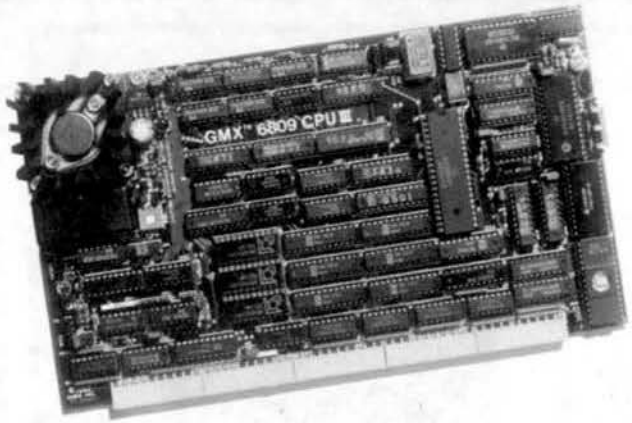
WESTCHESTER Applied Business Systems
Post Office Box 197, Briarcliff Manor, N.Y. 10510

All software is written in macro/assembler and runs under 6809 FLEX O/S. Terms: Check, Money Order, Visa or Mastercharge. Shipment first class. Add P&H $2.50 ($7.50 Foreign). NY Res add sales tax. Specify 5" or 8".

Sales: S. E. MEDIA, 1-800-338-6800, Consultation: 914-941-3552 (eves).

FLEX is a trademark of Technical Systems Consultants, Inc.

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.



GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.

For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

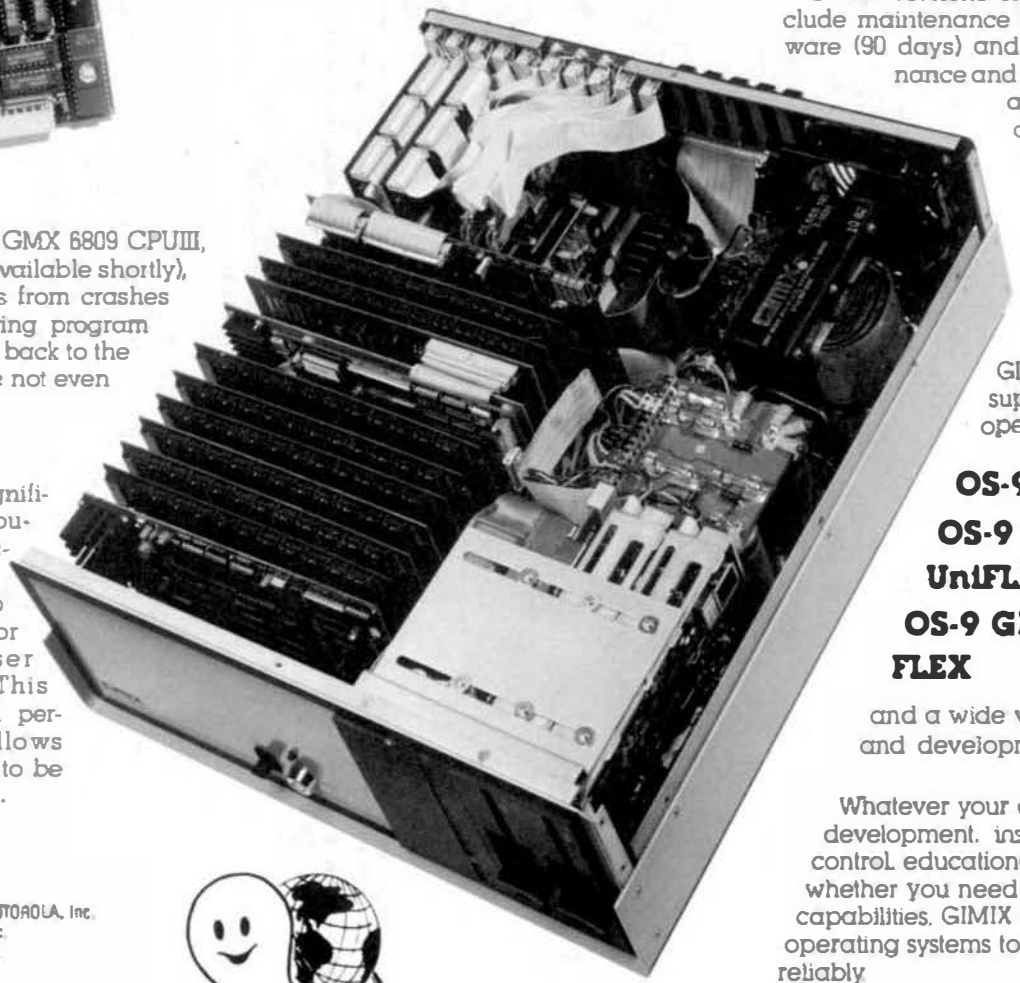
GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

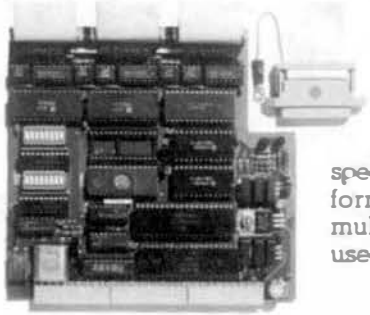
(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

'68' Micro Journal

For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs. e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.



The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, thereby freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

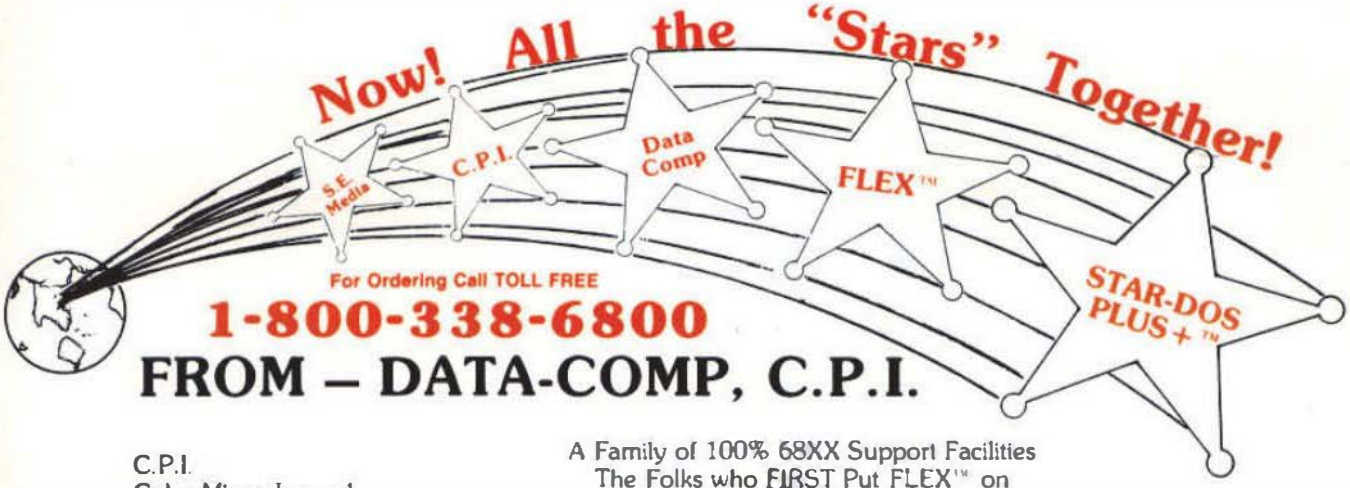
Whatever your application: software development, instrumentation, process control, educational, scientific or business, whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

Please phone or write if you need further information.

GIMIX INC.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

72



For Ordering Call TOLL FREE

1-800-338-6800

FROM – DATA-COMP, C.P.I.

C.P.I.
Color Micro Journal
'68' Micro Journal
Data-Comp
S.E. Media

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo
Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00
Only $79.95

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.50

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ Super 800 Support
+ Free Color Micro Journal 1 yr. sub.
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

DISK SYSTEMS FOR THE COLOR COMPUTER

THESE PACKAGES INCLUDE DRIVE, *CONTROLLER, POWER SUPPLY & CABINET, CABLE, AND MANUAL.

* SPECIFY WHAT CONTROLLER YOU WANT J&M. OR RADIO SHACK.

PAK #1 - 1 SINGLE SIDED, DOUBLE DENSITY SYS.	$389.95
PAK #2 - 2 SINGLE SIDED, DOUBLE DENSITY SYS.	$639.95
PAK #3 - 1 DOUBLE SIDED, DOUBLE DENSITY SYS.	$439.95
PAK #4 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS.	$699.95
PAK #5 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS. THINLINE DRIVES, HALF SIZE	$659.95
COLOR COMPUTER II 64K W/EXT. BASIC	$189.95

CONTROLLERS

J&M DISK CONTROLLER W/ JDOS OR RADIO SHACK DISK BASIC, SPECIFY WHAT DISK BASIC.	$139.95
RADIO SHACK DISK CONTROLLER 1.1	$134.95

DISK DRIVE CABLES

CABLE FOR ONE DRIVE	$ 19.95
CABLE FOR TWO DRIVES	$ 24.95

MISC

64K UPGRADE W/HOO. INSTRUCTIONS, C,D,E,F, AND COCO 2	$ 49.95
HJL KEYBOARDS	$ 69.95
MICRO TECH LOWER CASE ROM ADAPTER	$ 74.95
RADIO SHACK BASIC 1.2	$ 29.95
RADIO SHACK DISK BASIC 1.1	$ 29.95
RADIO SHACK EXT. BASIC	$ 39.95
SCREEN CLEAN CLEARS UP VIDEO DISTORTION	$ 39.95
MEMOREX DISKS 5" SS,DD	$ 24.00
SHIPPING INCLUDED ON DISK PRICES	
DISK DRIVE CABINET & POWER SUPPLY	$ 49.95
SINGLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$199.95
DOUBLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$249.95

PRINTERS

EPSON RX-80	$325.00
EPSON RX-80FT	$375.00
EPSON MX-100	$650.00
EPSON FX-100	$799.00
EPSON FX-80	$549.00
EPSON MX-70	$200.00

SERIAL BOARDS FOR PRINTERS

MX-SERIES	$119.95
FX-SERIES	$ 99.95

USA ADD 2% SHIPPING
FOREIGN ADD 5% SHIPPING

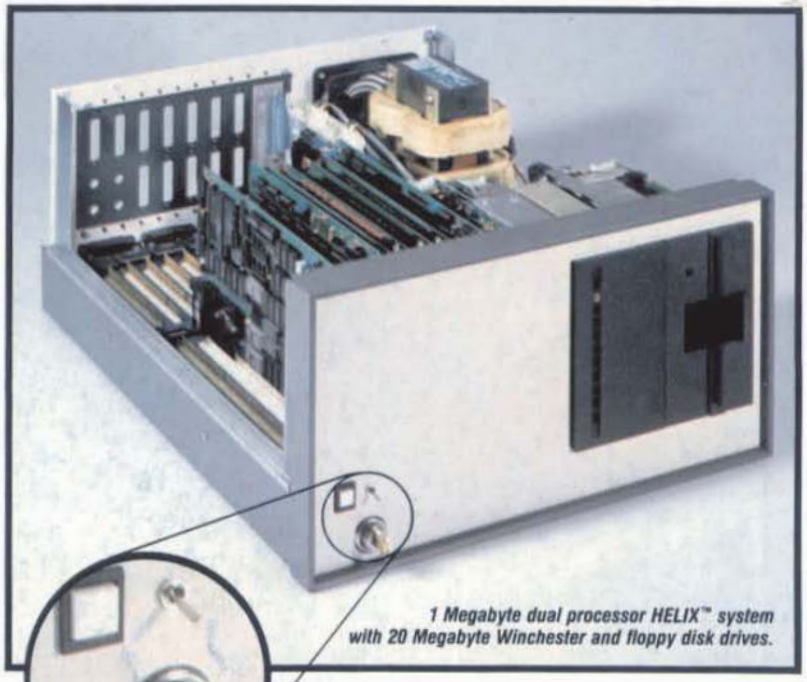
SPECIAL MX-100 $550.00

*FLEX is a Trademark of Technical System Consultants
*STAR DOS + is a Trademark of STAR-Kits & Data-Comp

5900 Cassandra Smith Rd. Hixson, TN 37343

68 Micro Journal
ISSN 0194-5025
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Hixson, TN
And Additional Mailing Offices



1 Megabyte dual processor HELIX™ system with 20 Megabyte Winchester and floppy disk drives.

***ONCE AGAIN HAZELWOOD COMPUTER SYSTEMS** demonstrates its leadership in computer technology by delivering the only computer system capable of switching between either the 6809 or the 68000 processor. Switching is easily accomplished by a simple front panel toggle switch. The reason we can offer this exclusive feature now, is that when our proven 6809 processor board was designed several years ago, we had the foresight to include the bus controls that allow processor switching.*

Hazelwood Computer Systems is also proud to be the first S-50/S-64 bus manufacturer to license and deliver the OS9/68K Operating System from Microware Systems Corporation. OS9/68K is the 68000 version of the popular and powerful OS9 Operating System. Utilizing our proven MC-20 disk controller, OS9/68K can conveniently share a Winchester disk with OS9. Changing from 6809 to 68000 operation is as simple as switching processors and booting the new system from the Winchester disk.

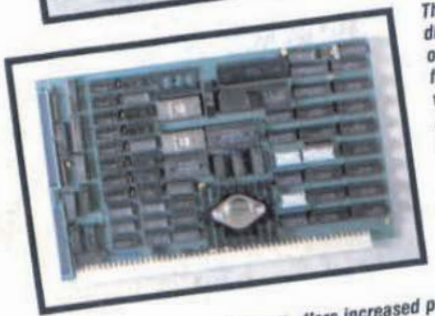
The ease of switching processors and operating systems makes a HELIX™ dual processor system the natural choice for software development. In addition, the advanced design of HELIX™ equipment, emphasizing performance and reliability, makes HELIX™ boards and systems the best value in computing offered anywhere.

System prices vary with configuration. Call for exact pricing.

THE SWITCH IS ON...

The CP-08 processor board utilizes a 68008 processor running at 10 Mhz clock rate. Using proprietary bus synchronization circuitry and single cycle DMA, the CP-08 achieves a marked performance increase over a 2 MHz 6809. Offering absolute compatibility with the 68000 instruction set, the 68008 addresses up to 1 Megabyte of memory. Also included on the CP-08 are up to 4K of ROM, an interrupt timer, and, with battery backup operation, a clock/calendar and 2K RAM. Implemented as a standard S-50 board, the CP-08 brings 68000 operation to S-50 bus computers.

ORDER: CP-08 **PRICE: $595**



The MC-20 Mass Storage Controller board interfaces up to 4 floppy and 8 Winchester disk drives to the S-50/S-64 bus. The MC-20 is an intelligent controller with its own 2 Mhz 6809 processor and 56K RAM. It provides DMA data transfers to a full 24 bit address. All disk operation requests are by logical block number, with the controller performing the necessary track/sector address calculations. Any combination of 5¼ or 8 inch floppy drives can be accommodated with all drive parameters, such as write precompensation, software controlled for each individual drive. Winchester drives are connected via a SASI bus interface. Block address mapping is provided which allows a single drive to be segmented into several logical units. The MC-20 is the controller of the MS-20 Mass Storage Subsystem which includes a 20 Megabyte Winchester drive.

ORDER: MC-20 **PRICE: $695**

OS9/68K

OS9/68K offers increased performance and larger user memory space while retaining all of the features of OS9. Disk file compatibility and operational similarity assures that present OS9 users can easily transfer their operations to the 68000. Included are an editor, assembler, linker, and debugger. A C compiler is available now. BASIC09 and other languages will be avilable soon.

ORDER: OS9/68K **PRICE: $250**

All items availeble stock to 30 days.
Prices subject to change without notice

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366, **314-281-1055**

OS9 and OS9/68K are registered trademarks of microware Systems Corp. HELIX is a trademark of Hazelwood Computer Systems.

HELIX™